Harmonious Hilbert curves
and other extradimensional space-filling curves[*]

Herman Haverkort[†]

November 2, 2012

Abstract

This paper introduces a new way of generalizing Hilbert's two-dimensional space-filling curve to arbitrary dimensions. The new curves, called *harmonious Hilbert curves*, have the unique property that for any $d' < d$, the d-dimensional curve is compatible with the d'-dimensional curve with respect to the order in which the curves visit the points of any d'-dimensional axis-parallel space that contains the origin. Similar generalizations to arbitrary dimensions are described for several variants of Peano's curve (the original Peano curve, the coil curve, the half-coil curve, and the Meurthe curve). The d-dimensional harmonious Hilbert curves and the Meurthe curves have neutral orientation: as compared to the curve as a whole, arbitrary pieces of the curve have each of $d!$ possible rotations with equal probability. Thus one could say these curves are 'statistically invariant' under rotation—unlike the Peano curves, the coil curves, the half-coil curves, and the familiar generalization of Hilbert curves by Butz and Moore.

In addition, prompted by an application in the construction of R-trees, this paper shows how to construct a $2d$-dimensional generalized Hilbert or Peano curve that traverses the points of a certain d-dimensional diagonally placed subspace in the order of a given d-dimensional generalized Hilbert or Peano curve.

Pseudocode is provided for comparison operators based on the curves presented in this paper.

1 Introduction

Space-filling curves A space-filling curve in d dimensions is a continuous, surjective mapping from $\mathbb{R}$ to $\mathbb{R}^d$. In the late 19th century Peano [15] described such mappings for $d = 2$ and $d = 3$. Since then, quite a number of space-filling curves have appeared in the literature, and space-filling curves have been applied in diverse areas such as spatial databases, load balancing in parallel computing, improving cache utilization in computations on large matrices, finite element methods, image compression, and combinatorial optimization [2].

Space-filling curves are usually recursive constructions that map the unit interval $[0, 1]$ to a subset of $\mathbb{R}^d$ that has measure larger than zero. In the case of Peano's two-dimensional curve, the subset of $\mathbb{R}^d$ that is 'filled' is the unit square. Peano's curve is based on subdividing this unit square into a grid of 3×3 square cells, and simultaneously subdividing the unit interval into nine subintervals. Each subinterval is then matched to a cell; thus Peano's curve traverses the cells one by one in a particular order. The mapping from unit interval to unit square is refined by applying the procedure recursively to each subinterval-cell pair, so that within each cell, the curve makes a similar traversal (see Figure 1(a)). The result is a fully-specified mapping from the unit interval to the unit square. A mapping from $\mathbb{R}$ to $\mathbb{R}^2$ could then be constructed by inverting the recursion, recursively considering the unit interval and the unit square as a subinterval and a cell of a larger interval and a larger square. A higher-dimensional version would be based on subdividing a d-dimensional hypercube into a grid of 3^d hypercubic cells.

[†]Dept. of Computer Science, Eindhoven University of Technology, the Netherlands, cs.herman@haverkort.net
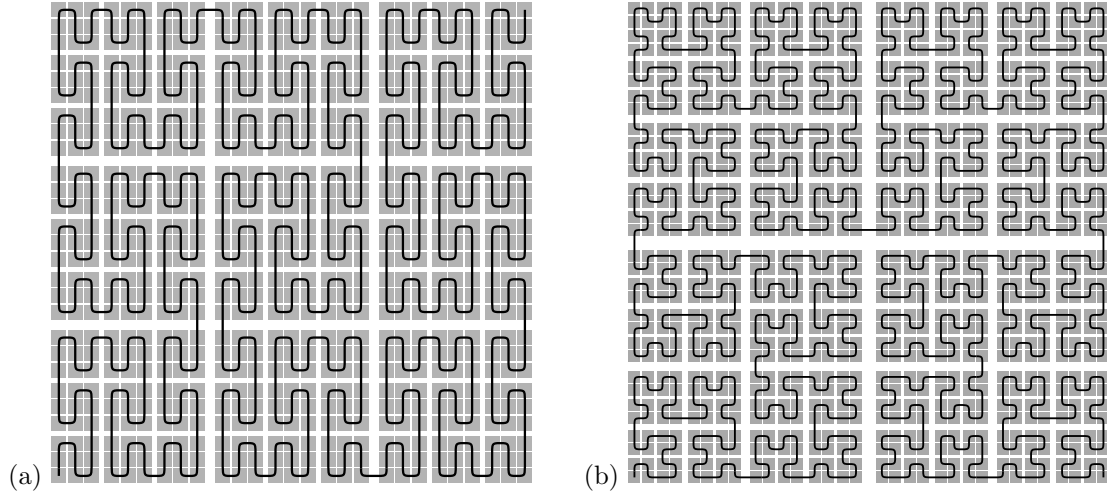
Figure 1: (a) A sketch of Peano's space-filling curve. (b) A sketch of Hilbert's space-filling curve.

Peano's curve has been the curve of choice for certain applications. However, in many other applications preference is given to curves based on subdividing squares (or hypercubes) into only 2^d squares (or hypercubes). The cell in which a given point p lies can then be determined by inspecting the binary representations of the coordinates of p bit by bit, and no divisions by three need to be computed. In response to Peano's publication, Hilbert [10] described a two-dimensional space-filling curve based on subdividing a square into four squares (Figure 1(b)). However, he did not describe how to do something similar in higher dimensions. It is now well-established that it can be done; in fact there are many ways to define a three-dimensional curve based on subdividing a cube into eight octants [1, 6].

The goal: extradimensional space-filling curves In this paper we consider different ways of generalizing Hilbert curves, Peano curves and similar curves to higher dimensions. Let P be a set of points on a d'-dimensional facet U' of a d-dimensional unit hypercube U, such that the origin is a vertex of U and U'. In this paper, we study generalizations of space-filling curves that have the following property: regardless of our choice of P and U', the order in which the points of P are visited by the d-dimensional curve that fills U, is the same as the order in which these points are visited by the d'-dimensional curve that fills U'.

To get some intuition for what this means, consider Figure 2. Figure 2(a) shows the order in which the two-dimensional Peano curve traverses the nine subsquares of the unit square, and Figure 2(b) shows the order in which a three-dimensional Peano curve traverses the 27 subcubes of the unit cube. If we consider only the subcubes that appear on the front, left, or bottom face of the unit cube, then we see that on each face, these subcubes are visited in the order of the two-dimensional Peano curve. Figure 3(a) shows the order in which the two-dimensional Hilbert curve traverses the four subsquares of the unit square, and Figure 3(b) shows the order in which the three-dimensional generalization by Moore [14] (citing Butz [3]) traverses the eight subcubes of the unit cube. Again, we see that on each face, the subcubes are visited in the order of the two-dimensional curve. However, from these observations one cannot conclude that all points of the front, left or bottom face are visited in the order of the two-dimensional curve. Figure 4 shows the next level of refinement of the two-dimensional Hilbert curve and the three-dimensional Butz-Moore curve. The order in which the subcubes of the front, left or bottom face are visited no longer corresponds to the two-dimensional curve. Our goal in this paper is to identify generalizations of the Hilbert curve and several variants of the Peano curve, in which the correspondence is maintained for any level of refinement of the order in which squares and cubes are visited.

To make this statement more precise, we should first define the order in which points appear along the curve more precisely. Let f be a space-filling curve, or more precisely, a continuous surjective function from $[0, 1]$ to a unit hypercube U. Let a *valid ordering* $\overline{f}$ of f be any bijection from the unit hypercube U to $[0, 1]$, such that for any point $p \in P$ we have $f(\overline{f}(p)) = p$.

Let $\mu : \{0, ..., d' - 1\} \rightarrow \{0, ..., d - 1\}$, $d > d'$ be a strictly increasing injection, that is, for any $i, j \in \{0, ..., d' - 1\}$ with $i < j$ we have $\mu(i) < \mu(j)$. This function μ will be used as a function that
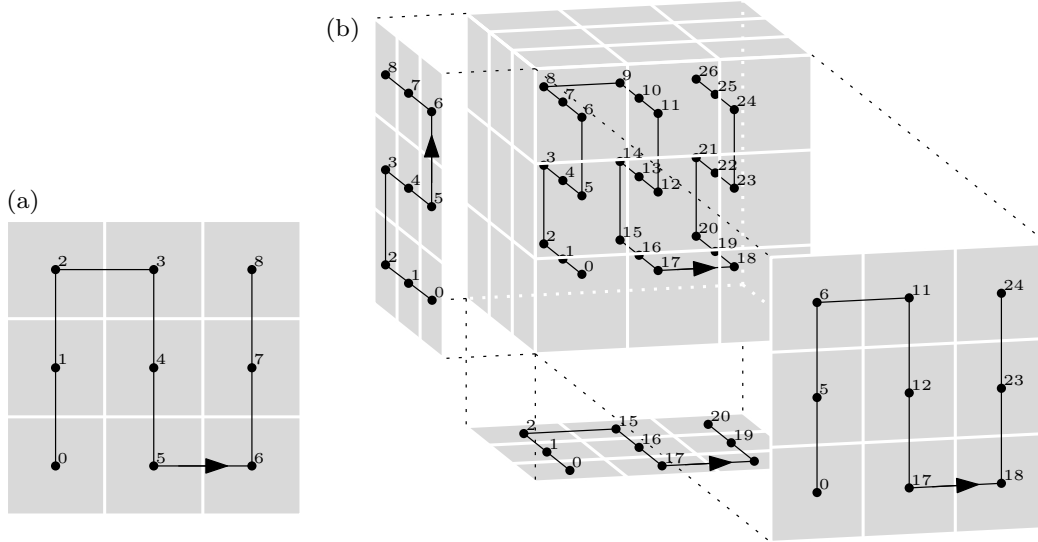
Figure 2: The Peano curve in two and three dimensions. (a) Ordering of nine squares in 2D. (b) Ordering of 27 cubes in 3D, with the induced orderings on the front, left, and bottom side.
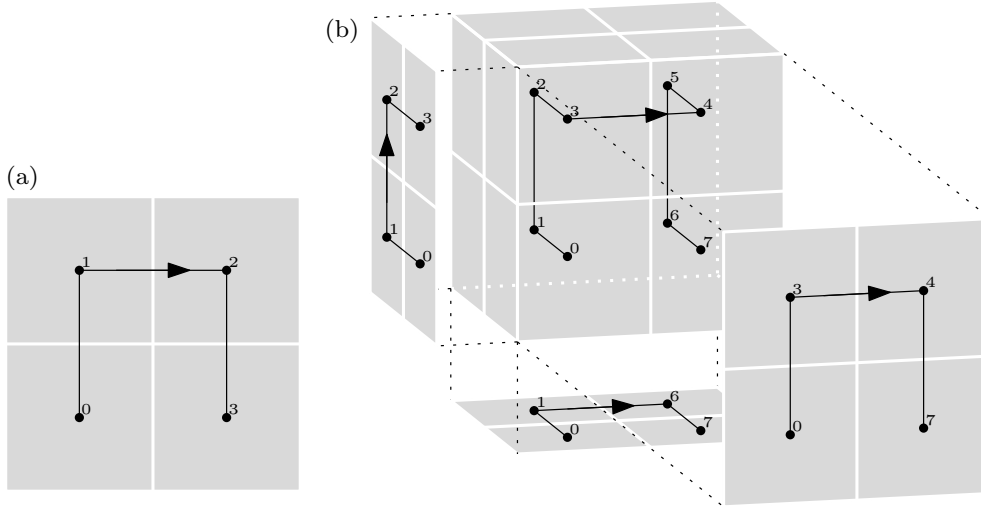

Figure 3: The Hilbert curve and the three-dimensional version of Butz-Moore. (a) Ordering of four squares in 2D. (b) Ordering of eight cubes in 3D, with the induced orderings on the front, left, and bottom side.

selects d' dimensions out of d, in the following way: given a d'-dimensional point $p = (p_0, ..., p_{d'-1})$, we define $\text{lift}_\mu(p)$ as the d-dimensional point $(q_0, ..., q_{d-1})$ such that $q_j = p_i$ when $j = \mu(i)$, and $q_j = 0$ when there is no i such that $j = \mu(i)$. In other words, $\text{lift}_\mu(p)$ is obtained from p by inserting zero coordinates at the positions not selected by μ. For a set of points P, let $\text{lift}_\mu(P)$ be the set $\bigcup_{p \in P} \text{lift}_\mu(p)$.

Definition 1 *A d-dimensional space-filling curve f is* extradimensional *to a d'-dimensional space-filling curve f' with co-domain U', if the following holds for any valid ordering $\overline{f'}$ of f' and any choice of μ: there is a valid ordering $\overline{f}$ of f such that for any pair of points $a, b \in U'$ we have $\overline{f'}(a) < \overline{f'}(b)$ if and only if $\overline{f}(\text{lift}_\mu(a)) < \overline{f}(\text{lift}_\mu(b))$.*

Definition 2 *An (infinite) set F of space-filling curves is* interdimensionally consistent, *if F contains a unique d-dimensional space-filling curve f_d for any integer $d \geq 1$, and $f_j \in F$ is extradimensional to $f_i \in F$ whenever $j > i$.*
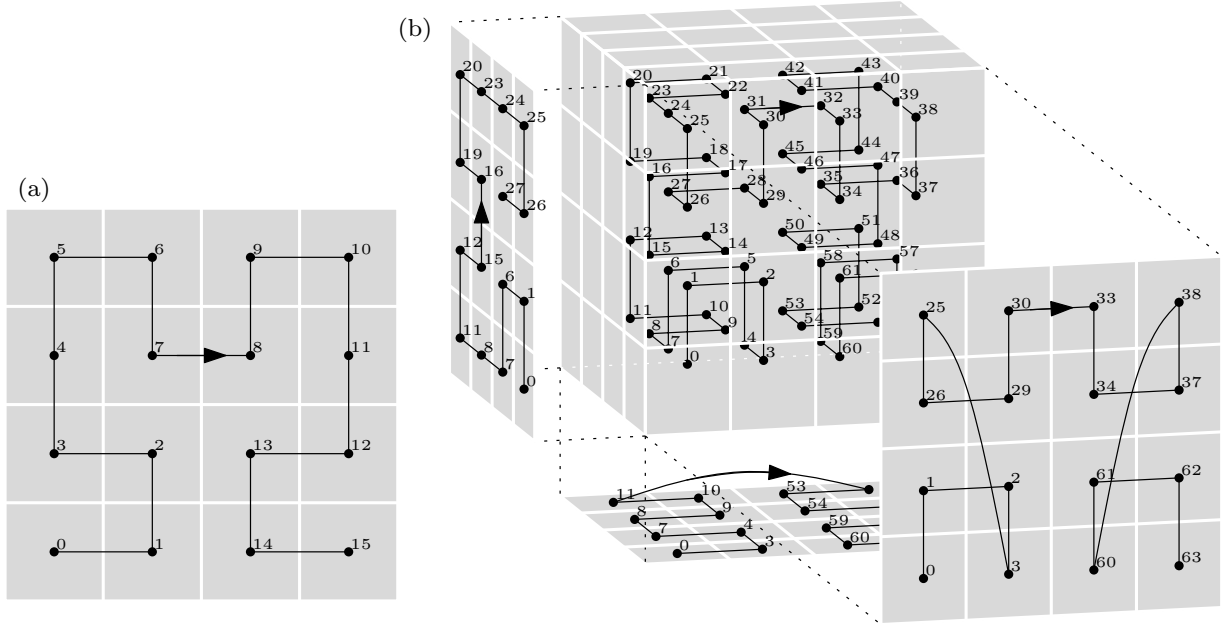
3

Figure 4: The Hilbert curve and the three-dimensional version of Butz-Moore. (a) Ordering of 16 squares in 2D. (b) Ordering of 64 cubes in 3D, with the induced orderings on the front, left, and bottom side.

Inverting the terminology, we could say that $f_{d'}$ is *intradimensional* to f_d if and only if f_d is extradimensional to $f_{d'}$, and each curve of an interdimensionally consistent set could be regarded as an intradimensional curve of an infinitely-dimensional[1] 'curve' f_∞.

In the definition of extradimensionality, a d-dimensional point is constructed from a d'-dimensional point by inserting zero coordinates at the positions not selected by μ. We will also consider *first-half-extradimensionality*, where a $2d$-dimensional point is constructed by appending d zeros to the coordinates of a d-dimensional point, and *diagonal-extradimensionality*, where a $2d$-dimensional point is constructed by concatenating two copies of the coordinate sequence of a d-dimensional point. In our previous work we found that first-half- and diagonal-extradimensional space-filling curves can be useful in improving the performance and robustness of certain data structures based on space-filling curves [9].

The results This work provides descriptions of interdimensionally consistent generalizations of Hilbert's curve and several variants of Peano's curve (coil, half-coil, and Meurthe) that caught my attention in previous work [7, 8] because of their excellent locality-preserving properties. Furthermore, this paper provides a general technique to derive first-half- and diagonal-extradimensional curves from a given generalized Hilbert or Peano curve. This enables the extension of our previous work on R-trees for rectangles to higher dimensions [9].

It turns out that the new generalized Hilbert curves and the generalized Meurthe curves share another interesting property: they are, in some sense, statistically invariant under rotation. For an intuitive understanding of what this means, one could look at Figure 1. At the highest level, the Peano curve shows more vertical steps than horizontal steps, and in recursion, it stays that way. Thus the curve has a clear orientation, which can be recognized immediately even if one sees only a small piece of the curve. This could be a disadvantage if the recognizable orientation somehow interacts negatively with the data in an application of the curve. On the other hand, at the smallest depicted level of detail of the Hilbert curve, all possible orientations of the curve occur approximately equally often. Therefore, from the looks of a small piece of the Hilbert curve one could not tell if the curve as a whole has been rotated or not: we say the curve has *neutral orientation*. The harmonious Hilbert curves and the Meurthe curves maintain this property also in higher dimensions—unlike the Peano curves, coil curves, half-coil curves and the Butz-Moore generalization of the Hilbert curve.

---

[1] I have not explored where the idea of infinitely-dimensional curves would lead us. I believe they might constitute a new class of "monster curves" (to use Gardner's words [4]).

How to read this paper Below, in Section 2, we fill first see how exactly we can define and describe space-filling curves like Hilbert's and Peano's. We will distinguish *2-regular* curves, which are based on subdividing d-dimensional hypercubes into 2^d subcubes, and *3-regular* curves, which are based on subdividing d-dimensional hypercubes into 3^d subcubes. In Section 3 we see some basic results that will come in useful in proving the claimed properties of the curves presented in the sections that follow. In Section 4 we will present interdimensionally consistent generalizations of several 3-regular curves, and prove that among those, the generalized Meurthe curves have neutral orientation. We will also give pseudocode for comparison operators that decide which of any two given points comes first on the curve. The pseudocode may not be a recipe for highly optimized code in any particular programming language, but it may give some insight in how one may produce such code and, in addition, it reveals properties of the curves that would be difficult to see otherwise. In Section 5 we learn about the Butz-Moore generalization of Hilbert curves and about our harmonious Hilbert curves in detail. Again, pseudocode is included. Section 6 defines the concepts of first-half- and diagonal-extradimensionality in detail, and explains how, given any b-regular curve f (for $b \in \{2, 3\}$), one can construct a b-regular curve g that is first-half- and/or diagonal-extradimensional to f. The paper concludes with Section 7 with a comparison between the results from the different sections of this paper and with open questions for possible further research.

In principle, Sections 4, 5 and 6 are independent from each other. The reader who wants to learn quickly how to get interdimensionally consistent, first-half-extradimensional or diagonal-extradimensional curves, may read Section 2 and then skip to any of the Sections 4, 5 or 6 right away. However, the reader who wants to understand the proofs in these sections, is advised to read Sections 2, 3, 4 and 5 in order, because the proofs in these sections build up in complexity.

2 Describing space-filling curves

2.1 Graphical descriptions and their interpretation

In general, we can define an order (*scanning order*) of points in d-dimensional space as follows. We give a set of rules, each of which specifies (i) how to subdivide a region in d-dimensional space into subregions; (ii) what is the order of these subregions; and (iii) for each subregion, which rule is to be applied to establish the order within that subregion. We also specify a unit region, and we indicate what rule is used to subdivide and order it.

Figure 5(a) shows an example (the scanning order depicted corresponds to a section of the $\beta\Omega$-curve by Wierum [16]). Each rule is identified by a letter, and pictured by showing a unit square, its subdivision into quadrants, the scanning order of the quadrants (by a directed curve along the outer vertices of the quadrants), and the rules applied to the quadrants (by letters). Within each quadrant, there is a scaled copy of the *base pattern*: the curve along the outer vertices of the quadrants. Variations of the rules that consist of simply rotating or mirroring the order within quadrants are indicated by rotating or mirroring the pattern. Variations of rules that consist of reversing the order of cells within a quadrant are indicated by reversing the direction of the pattern and by an overscore above the letter identifying the rule[2]. Figure 5(b) shows how applying the rules recursively, starting with rule R, results in an order of the subquadrants within each quadrant. Combined with the order of the quadrants, this gives an order of all squares in a 4×4 grid, which is sketched by the curve shown in Figure 5(c). From the orientations of the base patterns in Figure 5(b) and (c), one already gets a hint of the order of all squares in a 8×8 grid. By expanding the recursion further, one may order the cells of an arbitrarily fine grid.

The polygonal curve that connects the centres of these cells in order thus forms an arbitrarily fine approximation of the space-filling curve, which is more precisely defined as follows. At any level of recursion, let the grid consist of the cells $C_1, ..., C_n$, indexed from 1 to n according to the scanning

---

[2]Reversals are among the standard transformations applied in the definitions of space-filling curves in the literature—see, for example, the Gosper flowsnake [4], arguably one of the prototypical space-filling curves along with the curves by Peano, Hilbert, Lebesgue [12] and Sierpiński [2]. In fact, reversals are so standard, that they are often not described explicitly, and it is left to the reader to infer the reversals from how the pieces of the curve are put together. I prefer to be explicit about reversals. To indicate a reversal, it is necessary to reverse the direction of the pattern *and* to include the overscore. Figure 5(a) illustrates this. Rule J is asymmetric because different rules are applied to the first and the last quadrant. It is therefore important to distinguish between reflected, unreversed applications of J, as in the lower left quadrant of rule R, and unreflected, reversed applications of J, as in the lower right quadrant. Since the base pattern of J is symmetric, it is only the overscore that indicates which of these alternatives is meant.
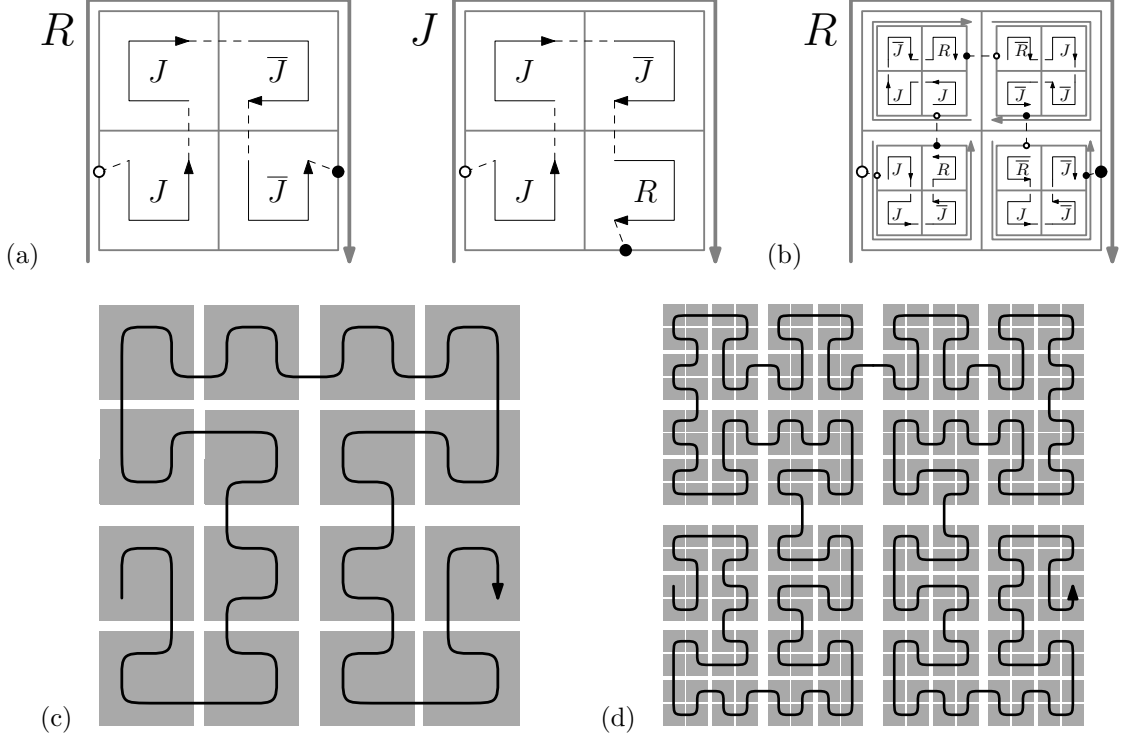
Figure 5: (a) Definition of a scanning order corresponding to a variant of the $\beta\Omega$-curve. (b) Applying the rules recursively within R. (c) The resulting order of the 4×4-grid. (d) Order of the 16×16-grid. On each level, the last cell of each quadrant shares a vertex (and even an edge) with the first cell of the next quadrant.

order, let $\text{vol}[1, i] = \sum_{j=1}^{i} \text{vol}(C_i)$ be the total volume of cells C_1 up to and including C_i, and define $\text{fraction}[1, i] = \text{vol}[1, i]/\text{vol}[1, n]$. Then the curve f maps the interval $[\text{fraction}[1, i-1], \text{fraction}[1, i]]$ to C_i, for all $i \in \{1, ..., n\}$. The image $f(q)$ of any point or interval q in $[0, 1]$ can now be obtained by taking the limit of the image of the intervals containing q as the level of recursion goes to infinity. Thus the definition of a scanning order is also a definition of a space-filling curve.

In the definitions of space-filling curves, the letters indicating the rules may be omitted if there are no reversals and no two rules have the same base pattern modulo rotation and reflection—see, for example, Figure 6. Figures 7, 8, 9 and 10 and Inset 1 show further examples of definitions of two- and three-dimensional space-filling curves.

Entrance and exit gates When the scanning order is refined to an infinitely fine grid, the first and the last cell visited shrink to points. These points are the images of 0 and 1 of the space-filling curve defined by the scanning order, and they typically lie on the boundary of the unit region[3]. We will call $f(0)$ the *entrance gate* and we will call $f(1)$ the *exit gate* of the curve; in Figure 5 their locations are indicated by the white dots and the black dots, respectively. Note that these dots are only shown for clarity: the locations of the gates are a consequence of the definition of the scanning order, rather than a part of the definition.

Using space-filling curves For many practical applications of space-filling curves it is not necessary to actually draw a curve. It is often enough to be able to decide for any two given points p and q, which of these appears first on the curve—that is, which point comes first in the scanning order. This can be decided by expanding the recursion only to the smallest depth at which p and q lie in different quadrants. A technical problem with this is that, down from some depth of recursion, a given point p may always

---

[3]It is not automatic that these points lie on the boundary, but space-filling curves are usually designed such that $f(0)$ and $f(1)$ do lie on the boundary—otherwise the curve cannot be continuous from one subregion into another subregion if each subregion is filled with a scaled-down copy of the curve.

lie on the boundary of two or more quadrants. This may create ambiguity about which of two given points comes first. This ambiguity can be resolved by using a consistent tie-breaking rule. For example, in the pseudocode in this paper, we will always assign each point p to the quadrant that lies to its upper right—that is, for a space-filling curve f we define $\overline{f}(p)$ as the common limit of the values t such that $f(t) = p + (\varepsilon, ..., \varepsilon)$ as $\varepsilon > 0$ approaches zero.

2.2 Classes of space-filling curves considered in this paper

There are many different possible generalizations of Peano's and Hilbert's curves to three and more dimensions. To be able describe the classes of curves we will be studying in this paper precisely, I will now introduce some terminology.

All space-filling curves studied in this paper are defined by a rule system as described above.

We will call a curve *b-regular* if the unit region of each rule is a d-dimensional unit hypercube, and each rule subdivides this hypercube into b^d smaller hypercubes of equal size. We adopt the convention that the unit hypercube is an axis-parallel hypercube that spans the interval $[0, 1]$ in each dimension; thus, each of the smaller hypercubes has width $1/b$.

We will call a curve a *mono-curve* if the defining rule system contains only a single rule.

We will call a curve *order-preserving* if it is a mono-curve and the transformations of the single defining rule within the subregions are restricted to rotation and reflection—excluding reversal of the order in which the subregions are traversed.

We will call a scanning order and the space-filling curve defined by it *vertex-continuous* if it has the following property: whenever the scanning order visits a set of hypercubes $C_1, ..., C_n$, in that order, and we refine the scanning order in each of these hypercubes recursively to depth ℓ so that we obtain an order on $n \cdot b^{\ell d}$ cells, for any $\ell \geq 1$, then the last cell within C_{i-1} shares at least one point with the first cell within C_i, for all $1 < i \leq n$ (see, for example, Figures 1 and 5(d)). Note that if a scanning order is not vertex-continuous, then it does not directly define a *continuous* mapping from the unit interval to a higher-dimensional space, and thus, the scanning order does not actually define a space-filling *curve*.[4] One of our main tasks in the following sections of this paper, will always be to prove that whatever is presented as a space-filling curve, is indeed vertex-continuous.

Inset 1 shows some of the variety of vertex-continuous 2-regular mono-curves that exist.

2.3 Numerical descriptions

Obviously, the graphical way of defining space-filling curves described above does not extend well to curves in more than three dimensions. Therefore, I will now introduce a concise numerical way of defining space-filling curves, tailored to b-regular order-preserving curves.

In the numerical notation, we will use d-dimensional coordinate vectors and d-digit numbers. We use subscripts from 0 to $d - 1$ to index the individual coordinates or digits, respectively. If x is a d-digit number, x_0 is the *most* significant and x_{d-1} is the *least* significant digit of x. By $(a)_d$ we denote the number that consists of d times the digit a.

The *rank* of each subregion is a base-b number from $\{0, ..., (b-1)_d\}$ that indicates the position of the subregion in the sorted order of subregions according to the rule that defines the curve: the first subregion has rank 0 and the last subregion has rank $(b-1)_d$. By $S(r)$ we denote the subregion with rank r. For ease of notation we write $\check{r}$ for $r - 1$, and we write $\hat{r}$ for $r + 1$.

The *location* of $S(r)$ is denoted by $c(r)$, which is a d-digit base-b number indicating that the subregion's corner that is closest to the origin is located at $\frac{1}{b}(c_0(r), ..., c_{d-1}(r))$ and the subregion's corner that is farthest from the origin is located at $\frac{1}{b}(c_0(r) + 1, ..., c_{d-1}(r) + 1)$.

The curve within each subregion $S(r)$ can be obtained from the curve in the unit hypercube as a whole by first rotating it, then reflecting it, then scaling it with a factor $1/b$ in all dimensions, and then translating it into place. I will now describe how rotations, reflections and translations are specified. The rotation is specified by a permutation $a(r)$, which is a d-digit base-d number of which the digits form a permutation of $\{0, ..., d - 1\}$. This permutation specifies which axis $a_i(r)$ of the curve through the unit
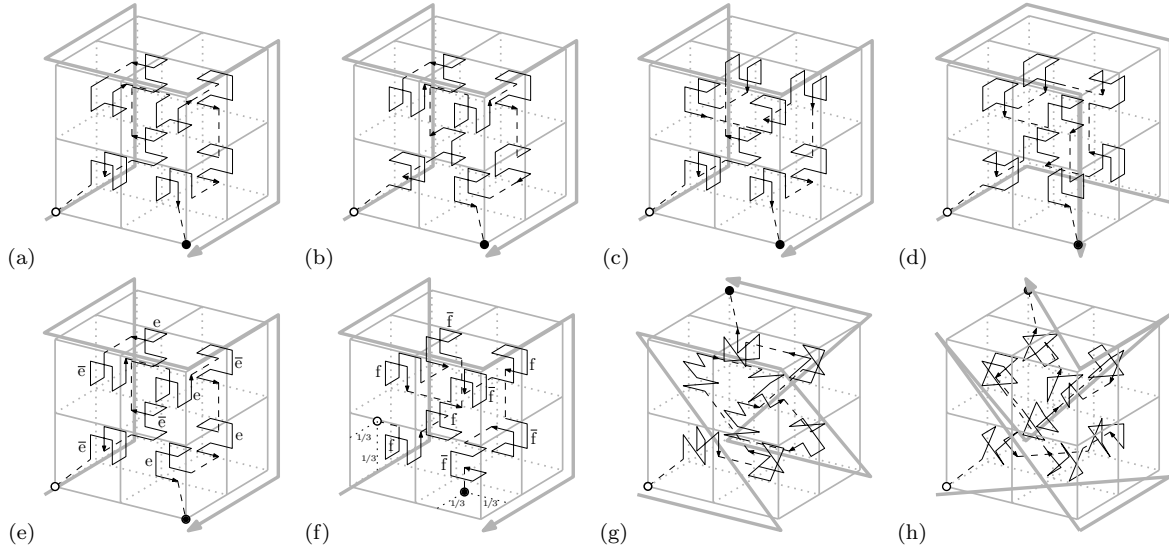
---

[4]A work-around would be to insert infinitely many connecting line segments to bridge the gaps. In effect, this is how Lebesgue makes a space-filling curve out of the non-vertex-continuous scanning order that is nowadays known as Z-order [12]. However, such tricks come at the expense of decreased performance in applications [8], and therefore they are not considered in this paper.

hypercube is rotated onto axis i within subregion r. By $\bar{a}(r)$ we denote the inverse of this permutation, that is, $\bar{a}_i(r) = j$ if and only if $a_j(r) = i$. The reflections are specified by a d-digit binary number $m(r)$, where $m_i(r) = 1$ if and only if after rotation, the curve within subregion r should be reflected in a plane orthogonal to axis i. The required translation vector $o(r)$ is implied by the rotation, the reflections and the location of the subregion, and does not need to be specified separately.

More precisely, $a(r)$ and $m(r)$ define a transformation matrix $M(r)$ with rows and columns numbered from 0 to $d-1$ and the following entries:

$$
\begin{aligned}
M(r)_{ij} = 1 \quad & \text{if } j = a_i(r) \text{ and } m_i(r) = 0; \\
M(r)_{ij} = -1 \quad & \text{if } j = a_i(r) \text{ and } m_i(r) = 1; \\
M(r)_{ij} = 0 \quad & \text{if } j \neq a_i(r).
\end{aligned}
$$

rank	loc.	perm.	refl.	exit
00	00	10	00	$\frac{1}{2}(0,1)$
01	01	01	00	$\frac{1}{2}(1,1)$
10	11	01	00	$\frac{1}{2}(2,1)$
11	10	10	11	$\frac{1}{2}(2,0)$

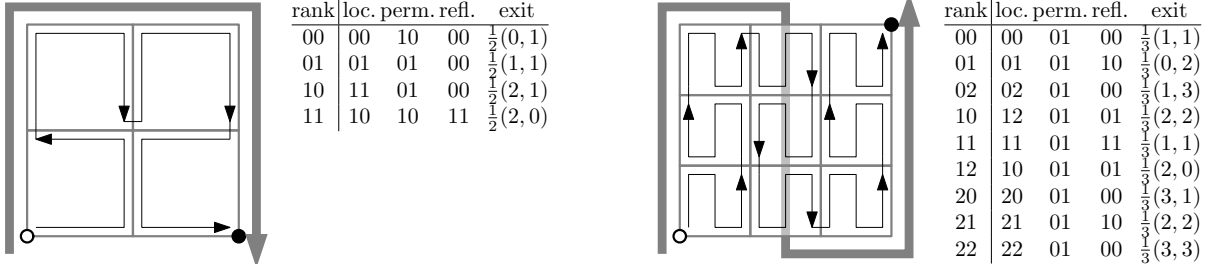| rank	loc.	perm.	refl.	exit
00	00	01	00	$\frac{1}{3}(1,1)$
01	01	01	10	$\frac{1}{3}(0,2)$
02	02	01	00	$\frac{1}{3}(1,3)$
10	12	01	01	$\frac{1}{3}(2,2)$
11	11	01	11	$\frac{1}{3}(1,1)$
12	10	01	01	$\frac{1}{3}(2,0)$
20	20	01	00	$\frac{1}{3}(3,1)$
21	21	01	10	$\frac{1}{3}(2,2)$
22	22	01	00	$\frac{1}{3}(3,3)$

Figure 6: Hilbert's curve (left) and Peano's curve (right) defined graphically and by means of a table.

By $\overline{M}(r)$ we denote the inverse of this matrix:

$$\begin{aligned}
\overline{M}(r)_{ji} &= 1 && \text{if } i = \overline{a}_j(r) \text{ and } m_i(r) = 0; \\
\overline{M}(r)_{ji} &= -1 && \text{if } i = \overline{a}_j(r) \text{ and } m_i(r) = 1; \\
\overline{M}(r)_{ji} &= 0 && \text{if } i \neq \overline{a}_j(r).
\end{aligned}$$

The transformation $\tau(r) : p \to \frac{1}{b}M(r)p + o(r)$ now maps each point p of the curve through the unit hypercube to a point in subregion $S(r)$. Recall that the space-filling curve is a function f from $[0,1]$ to the unit hypercube $[0,1]^d$. Considering the subregion with rank r, the function f maps the domain $[r/b^d, (r+1)/b^d]$ to $S(r)$. The transformation $\tau(r)$ describes the relation between, on one hand, f restricted to the domain $[r/b^d, (r+1)/b^d]$, and on the other hand, f on the domain $[0,1]$: for $x \in [r/b^d, (r+1)/b^d]$ we have $f(x) = \tau(r)(f(s_r(x)))$, where $s_r(x) = b^d x - r$ is the function that scales up the domain $[r/b^d, (r+1)/b^d]$ to $[0,1]$. Conversely, there is a valid ordering $\overline{f}$ of f such that for any point p in the interior of subregion r we have: $\overline{f}(p) = \overline{s_r}\left(\overline{f}(\overline{\tau}(r)(p))\right)$, where $\overline{\tau}(r)(p) = \overline{M}(r)b(p - o(r))$ and $\overline{s_r}(x) = (x + r)/b^d$.

A curve can now be described by a table that lists, for each subregion $S(r)$, the location $c(r)$, the permutation $a(r)$, the reflections $m(r)$, and, redundantly (if desired for clarity), the entrance and/or exit gates of each subregion. Figure 6 shows the definitions of the two-dimensional curves by Hilbert and Peano in graphical notation and in table format.

2.4 Implementing a comparison operator

We will know see how we can easily implement a comparison operator for an order-preserving b-regular curve. It is not necessary to read this section to be able to understand the descriptions and proofs in the rest of this paper, and it is not necessary to read this section to be able to use the pseudocode presented in the rest of this paper. However, it may help in understanding why the pseudocode in Sections 4–6 works.

Our goal in this section is the following: given two points p, q in $[0,1]^d$, whose coordinates are given as base-b numbers, decide whether $\overline{f}(p) < \overline{f}(q)$. As an ordering $\overline{f}$ corresponding to f we choose the ordering that assigns each point p to the subregion that lies above/behind/to the right of it in all dimensions—that is, we define $\overline{f}(p)$ as the common limit of the values t such that $f(t) = p + (\varepsilon, ..., \varepsilon)$ as $\varepsilon > 0$ approaches zero. This defines $\overline{f}$ only for points in $[0,1)^d$; it is not defined for points of which one or more coordinates are equal to 1, and our pseudocode will not deal with such points. One could define $\overline{f}$ also for points with one or more coordinates equal to 1, and one could easily adapt the algorithms presented in this paper to handle such coordinates, by representing the number 1 as 0.111... (if $b = 2$) or 0.222... (if $b = 3$).

The idea of our basic algorithm (Algorithm 2.1) is as follows. The algorithm takes two points p and q, each given as a set of d coordinates $(p[0], ..., p[d-1])$ and $(q[0], ..., q[d-1])$, respectively. Assume that each coordinate is in $[0,1)$ and it is given as the string of digits that constitute the fractional part of its representation as a base-b number. The algorithm first removes the first digit of the fractional part of each coordinate. This is done with a function EXTRACTFIRSTDIGIT(x), which removes the first digit from the digit string x and returns the removed digit—if EXTRACTFIRSTDIGIT is called on an empty string, it simply returns a zero. Now let c_p and c_q be the concatenated removed digits of p and q, respectively. Determine r_p and r_q such that $c(r_p) = c_p$ and $c(r_q) = c_q$; thus r_p and r_q are the ranks of the regions containing p and q, respectively. If $r_p < r_q$, return true; if $r_p > r_q$, return false; otherwise, we

apply the transformation $x \to \overline{M}(r_p)x$ to the points p and q that remained after removing the first digits of their coordinates, and recurse on the resulting points p and q. In fact, in Algorithm 2.1 we do not apply the transformation right away, but instead we maintain a transformation matrix T which is the composition of all transformations $\overline{M}(r_p)$ that should have been applied at all higher levels of recursion. We apply this composed transformation T only to the digits extracted, on Line 5.

Algorithm 2.1: General comparison operator for a b-regular curve: the main idea

Input: Points $p = (p[0], ..., p[d-1])$ and $q = (q[0], ..., q[d-1])$, with coordinates in $[0, 1)$, given by the digits of the fractional part in base-b representation

Output: **true** if $\overline{f}(p) < \overline{f}(q)$, **false** otherwise

1   $T \leftarrow$ identity matrix of size d
2   **repeat**
3   **for** $i \leftarrow 0$ **to** $d-1$ **do**
4         $c_p[i] \leftarrow$ EXTRACTFIRSTDIGIT$(p[i])$; $c_q[i] \leftarrow$ EXTRACTFIRSTDIGIT$(q[i])$
5      $c_p \leftarrow T \cdot c_p$; $c_q \leftarrow T \cdot c_q$
6   add $b-1$ to all elements of c_p and c_q that are negative
7   compute r_p, r_q such that $c(r_p) = c_p$ and $c(r_q) = c_q$
8   **if** $r_p < r_q$ **then return true else if** $r_p > r_q$ **then return false**
       // $c_p = c_q$ and $r_p = r_q$
9      $T \leftarrow \overline{M}(r_p) \cdot T$
10 **until** *all remaining digit strings $p[0], ..., p[d-1]$ and $q[0], ..., q[d-1]$ are empty*
11 **return false**   // p and q are equal

The correctness of Algorithm 2.1 is based on the following observations. To compare $\overline{f}(p) = \overline{s_r}\left(\overline{f}(\overline{\tau}(r)(p))\right)$ and $\overline{f}(q) = \overline{s_r}\left(\overline{f}(\overline{\tau}(r)(q))\right)$ we do not have to apply $\overline{s_r}$ if both points lie in the same region $S(r)$: we can simply compare $\overline{f}(\overline{\tau}(r)(p))$ to $\overline{f}(\overline{\tau}(r)(q))$. Algorithm 2.1 does this as follows. Removing the first digits of the coordinates implements the transformation $x \to b(x - o'(r))$, where $o'(r)$ is the corner of $S(r)$ closest to the origin. Thus, assuming $o(r) = o'(r)$, subsequently applying $x \to \overline{M}(r)x$ implements $\overline{\tau}(r)$. If there are reflections, that is, negative entries in $\overline{M}(r)$, then the situation is slightly more complicated. The translation vector $o(r)$ should be the coordinates of the corner of region $S(r)$ that maps to the origin under the transformation $\overline{\tau}(r)$. Mapping $o'(r)$ to the origin instead, results in a translation of the unit hypercube $\overline{\tau}(r)(S(r))$ over a distance -1 in each of the reflected coordinates. Line 6 corrects this by adding $b-1$ to each extracted digit of a negative coordinate—which amounts to adding $0.111... = 1$ (if $b = 2$) or $0.222... = 1$ (if $b = 3$).

Algorithm 2.2 shows a version of Algorithm 2.1 which is more explicit about how to maintain and apply the transformation matrix T. Note that T is always a matrix with exactly one non-zero entry in each row, and this entry is either 1 or -1. Therefore, instead of maintaining a complete matrix, we can simply maintain where the non-zero entries are and whether they are 1 or -1.

2.5 Reflected Gray codes

Many variants of Peano's and Hilbert's curves are based on *Gray codes*: sequences of base-b numbers of d digits such that each number from $\{0, ..., b^d - 1\}$ occurs exactly once and consecutive numbers differ in only one digit. More precisely, we require that the difference in this digit is always exactly one. When the numbers of such a Gray code are interpreted as the locations $c(r)$ of the regions $S(r)$ for $r = 0, ..., b^d - 1$, the result is a space-filling curve in which each visited region $S(r)$ shares a $(d-1)$-dimensional face with its predecessor $S(\check{r})$ (recall that we use $\check{r}$ to denote $r-1$); for examples, see Figures (a)–(f) in Inset 1. Thus, each region fits snugly together with the previous region. This tends to result in space-filling curves with good locality-preserving properties, although by itself, it is not a guarantee [6].

In many of the definitions of space-filling curves that follow, we will use a specific type of Gray codes called *reflected Gray codes*. By $b\text{RGC}^d$ we denote the *reflected Gray code* with base b. This code is a sequence of b^d base-b numbers, containing each of the b^d different base-b numbers exactly once. The code is defined recursively as follows. Let $a \oplus b\text{RGC}^d$ denote the sequence that is obtained by prefixing each number of $b\text{RGC}^d$ with the digit a, let $\text{rev}(b\text{RGC}^d)$ denote $b\text{RGC}^d$ in reverse order, and let $\text{revifodd}(a, b\text{RGC}^d)$ be

---

Algorithm 2.2: General comparison operator for a b-regular curve: more detailed algorithm

---

Input: Points $p = (p[0], ..., p[d-1])$ and $q = (q[0], ..., q[d-1])$, with coordinates in $[0, 1)$, given by
   the digits of the fractional part in base-b representation

Output: **true** if $\overline{f}(p) < \overline{f}(q)$, **false** otherwise

---

1 **for** $i \leftarrow 0$ **to** $d-1$ **do** $reflected[i] \leftarrow$ **false**   // maintains which columns of T have minus
2 **for** $i \leftarrow 0$ **to** $d-1$ **do** $axis[i] \leftarrow i$ // maintains which entry in each row of T is non-zero
3 initialize an array $altAxis[0..d-1]$ // for a new value of $axis$ under construction
4 **repeat**
5   **for** $i \leftarrow 0$ **to** $d-1$ **do**
6          $c_p[i] \leftarrow \text{EXTRACTFIRSTDIGIT}(p[axis[i]]); c_q[i] \leftarrow \text{EXTRACTFIRSTDIGIT}(q[axis[i]])$
7   **if** $reflected[axis[i]]$ **then**
8             $c_p[i] \leftarrow (b-1-c_p[i]); c_q[i] \leftarrow (b-1-c_q[i])$
9   compute r_p, r_q such that $c(r_p) = c_p$ and $c(r_q) = c_q$
10   **if** $r_p < r_q$ **then return true else if** $r_p > r_q$ **then return false**
     // $c_p = c_q$ and $r_p = r_q$
11   **for** $i \leftarrow 0$ **to** $d-1$ **do**
12          $j \leftarrow \overline{a}_i(r_p)$
13          $altAxis[i] \leftarrow axis[j]$
14   **if** $m_j(r_p) = 1$ **then** $reflected[altAxis[i]] \leftarrow \neg reflected[altAxis[i]]$
15   swap $axis$ and $altAxis$
16 **until** *all remaining digit strings $p[0], ..., p[d-1]$ and $q[0], ..., q[d-1]$ are empty*
17 **return false**   // p and q are equal

$b\text{RGC}^d$ if a is even, and $\text{rev}(b\text{RGC}^d)$ if a is odd. Then $b\text{RGC}^1$ is the sequence $\langle 0, 1, ..., b-1 \rangle$, and $b\text{RGC}^{d+1}$, for $d \geq 1$, consists of the concatenation of the sequences $a \oplus \text{revifodd}(a, b\text{RGC}^d)$ for $a = 0, 1, ..., b-1$ in order. For example, $3\text{RGC}^1 = \langle 0, 1, 2 \rangle$ and $3\text{RGC}^2 = \langle 00, 01, 02, 12, 11, 10, 20, 21, 22 \rangle$. We can index $b\text{RGC}^d$ by d-digit base-b numbers. More precisely: we can consider $b\text{RGC}^d$ to be a bijection between the d-digit base-b numbers and the d-digit base-b numbers, where $b\text{RGC}^d(r)$ is the element r (counting from zero) of the sequence $b\text{RGC}^d$. Observe that $b\text{RGC}^d(0) = 0$.

The reader may verify that by construction, two consecutive elements $b\text{RGC}^d(\check{r})$ and $b\text{RGC}^d(r)$ differ in only one digit, and the difference in value of this digit is one.

Furthermore, $b\text{RGC}^d(r)$ can be obtained from r by going through the digits of r in order from r_0 to r_{d-1}: whenever a digit r_i is encountered such that r_i is odd, we 'reflect' all following digits, that is, we replace r_j by $b-1-r_j$ for all $j > i$. To understand why this works, consider that after having seen r_i, we have fixed the first $i+1$ digits of $b\text{RGC}^d(r)$; if $b\text{RGC}_i^d(r)$ is even, then the remaining digits must constitute the element of $b\text{RGC}^{d-i-1}$ indexed by the base-b number $r_{i+1}...r_{d-1}$, whereas if $b\text{RGC}_i^d(r)$ is odd, then the remaining digits must constitute the element of $\text{rev}(b\text{RGC}^{d-i-1})$ indexed by the base-b number $r_{i+1}...r_{d-1}$, or equivalently, the element of $b\text{RGC}^{d-i-1}$ indexed by $r'_{i+1}...r'_{d-1}$, where $r'_j = b-1-r_j$.

Conversely, we compute r from $b\text{RGC}^d(r)$ with Algorithm 2.3. For a b-regular curve of which $c(r)$ is defined as $b\text{RGC}^d(r)$, we can integrate Algorithm 2.3 into Algorithm 2.2 (which computes which of two points comes first along a space-filling curve) and obtain Algorithm 2.4.

Algorithm 2.3: Gray-decoding algorithm for $b\text{RGC}^d$

Input: An element of $b\text{RGC}^d$, given as an array $c = (c[0], ..., c[d-1])$ of base-b digits

Output: $rank$ s.t. $b\text{RGC}^d(rank) = c$, given as an array $rank[0..d-1]$ of base-b digits

---

1 $forward \leftarrow$ **true**   // indicates whether we go through $b\text{RGC}^{d-i}$ forwards or backwards
2 **for** $i \leftarrow 0$ **to** $d-1$ **do**
3   **if** $forward =$ **true then** $rank[i] = c[i]$ **else** $rank[i] = (b-1-c[i])$
4   **if** $c[i]$ *is odd* **then** $forward \leftarrow \neg forward$

Algorithm 2.4: Comparison operator for a b-regular curve with $c(r) = b\text{RGC}^d(r)$

Input: Points $p = (p[0], ..., p[d-1])$ and $q = (q[0], ..., q[d-1])$, with coordinates in $[0, 1)$, given by the digits of the fractional part in base-b representation

Output: **true** if $\overline{f}(p) < \overline{f}(q)$, **false** otherwise

1 **for** $i \leftarrow 0$ **to** $d-1$ **do** $reflected[i] \leftarrow$ **false**   // maintains which columns of T have minus
2 **for** $i \leftarrow 0$ **to** $d-1$ **do** $axis[i] \leftarrow i$ // maintains which entry in each row of T is non-zero
3 initialize an array $altAxis[0..d-1]$ // for a new value of $axis$ under construction
4 **repeat**
5 $\quad$ $forward \leftarrow$ **true**   // indicates whether we go through $b\text{RGC}^{d-i}$ forwards or backwards
6 $\quad$ **for** $i \leftarrow 0$ **to** $d-1$ **do**
7 $\quad\quad$ $pDigit \leftarrow \text{EXTRACTFIRSTDIGIT}(p[axis[i]])$; $qDigit \leftarrow \text{EXTRACTFIRSTDIGIT}(q[axis[i]])$
8 $\quad\quad$ **if** $reflected[axis[i]]$ **then**
9 $\quad\quad\quad$ $pDigit \leftarrow (b - 1 - pDigit)$; $qDigit \leftarrow (b - 1 - qDigit)$
10 $\quad\quad$ **if** $pDigit < qDigit$ **then return** $forward$ **else if** $pDigit > qDigit$ **then return** $\neg forward$
11 $\quad\quad$ **if** $forward$ **then** $rank[i] \leftarrow pDigit$ **else** $rank[i] \leftarrow (b - 1 - pDigit)$
12 $\quad\quad$ **if** $pDigit$ is odd **then** $forward \leftarrow \neg forward$
13 $\quad$ **for** $i \leftarrow 0$ **to** $d-1$ **do**
14 $\quad\quad$ $j \leftarrow \overline{a}_i(rank)$
15 $\quad\quad$ $altAxis[i] \leftarrow axis[j]$
16 $\quad\quad$ **if** $m_j(rank) = 1$ **then** $reflected[altAxis[i]] \leftarrow \neg reflected[altAxis[i]]$
17 $\quad$ swap $axis$ and $altAxis$
18 **until** *all remaining digit strings $p[0], ..., p[d-1]$ and $q[0], ..., q[d-1]$ are empty*
19 **return false**   // p and q are equal

---

3   Proof ingredients

In this section we establish notation and basic results that are needed for the proofs in the following sections. Readers who wish to learn about the final results and pseudocode for implementations only without studying the proofs, may skip this section. More precisely, in Section 3.1 we set up notation common to most proofs; in Section 3.2 we set up basic terminology and observations which will allow us to reason about interdimensional consistency; in Section 3.3 we do the same for the concept of neutral orientation. In Section 3.4 we establish basic properties of reflected Gray codes that we will need in the interdimensional-consistency proofs; this last section can also be skipped, to be read, if desired, when it is referred to later on.

3.1   Notation for editing numbers, vectors and sequences

The take-out operator   If a is a vector or a number, we will use $\overset{i}{\times} a$ to denote the vector or number that results from removing the element or digit a_i and renumbering the remaining elements consecutively, starting from zero. Thus, if $b = \overset{i}{\times} a$, we have $b_j = a_j$ for $j < i$, and $b_j = a_{j+1}$ for $j \geq i$. When the take-out operator is applied to a permutation a, the remaining elements are numbered consecutively. Thus, we have:

$$\begin{array}{ll} \overset{i}{\times} a_j = a_j & \text{if } 0 \leq j < i \text{ and } a_j < a_i; \\ \overset{i}{\times} a_j = a_j - 1 & \text{if } 0 \leq j < i \text{ and } a_j > a_i; \\ \overset{i}{\times} a_j = a_{j+1} & \text{if } i \leq j < d-1 \text{ and } a_{j+1} < a_i; \\ \overset{i}{\times} a_j = a_{j+1} - 1 & \text{if } i \leq j < d-1 \text{ and } a_{j+1} > a_i. \end{array}$$

For example, if $a = 30142$, then $\overset{2}{\times} a = 2041$. The take-out operator can also be applied to sets and sequences: if A is a set or sequence, then $\overset{i}{\times} A$ is the set or sequence that results from applying $\overset{i}{\times}$ to every member of the set or sequence. For example, if A is the sequence of ternary numbers $\langle 010, 000, 210, 222 \rangle$, then $\overset{2}{\times} A$ is the sequence $\langle 01, 00, 21, 22 \rangle$.

The reduce operator If A is a set or sequence, then we use $\overset{i=j}{\times} A$ to denote the set or sequence that results from only selecting those elements $a \in A$ such that $a_i = j$, and applying $\overset{i}{\times}$ to those elements. For example, if A is the sequence of ternary numbers $\langle 010, 000, 210, 222 \rangle$, then $\overset{2=0}{\times} A$ is the sequence $\langle 01, 00, 21 \rangle$.

The reflection operator If a is a base-b number, then we use $\overset{i}{\ominus} a$ to denote the number a that results from replacing digit a_i by its complement with respect to $b - 1$. For example, if a is the ternary number 20, then $\overset{1}{\ominus} a = 22$, and if a is the binary number 1001, then $\overset{2}{\ominus} a = 1011$. Like the take-out operator, the reflection operator can also be applied to sets and sequences.

The insertion operator If a is a vector or a number, we will use $\overset{i=j}{\wedge} a$ to denote the vector or number that results from inserting an element or digit a_i with value j, and re-indexing the original elements or digits $a_i, ...$ as $a_{i+1}, ...$ For example, if a is the ternary number 120, then $\overset{2=1}{\wedge} a = 1210$. Like the take-out operator, the insertion operator can also be applied to sets and sequences.

3.2 Visible orders of space-filling curves

Recall from Definition 2 that an (infinite) set F of space-filling curves is *interdimensionally consistent*, if F contains a unique d-dimensional space-filling curve f_d for any integer $d \geq 1$, and $f_j \in F$ is extradimensional to $f_i \in F$ whenever $j > i$. Observe that, to show that a given set F of space-filling curves is interdimensionally consistent, we only need to show that each f_j is extradimensional to f_{j-1}. Then it follows by induction that f_j is also extradimensional to all f_i with $j > i$.

To make the proofs in the rest of this paper easier to read, we introduce the following terminology. The *front face* F_i^0 of a d-dimensional unit hypercube is the $(d-1)$-dimensional face that consists of the points $\{(q_0, ..., q_d) \in [0,1]^d \mid q_i = 0\}$, and the *back face* F_i^1 of a d-dimensional unit hypercube is the $(d-1)$-dimensional face that consists of the points $\{(q_0, ..., q_{d-1}) \in [0,1]^d \mid q_i = 1\}$. With a *face of a space-filling curve g* we mean a face of the unit hypercube filled by g.

The *visible order* of a valid ordering $\overline{g}$ on a face F_i^k of a d-dimensional space-filling curve g, is the function $\overline{f} : [0,1]^{d-1} \to [0,1]$ that is defined by the following two properties: (i) for any pair of points $a, b \in F_i^k$ we have:

$$\overline{f}(\overset{i}{\times} a) < \overline{f}(\overset{i}{\times} b) \Leftrightarrow \overline{g}(a) < \overline{g}(b),$$

and (ii) the measure of the set $\{p \mid \overline{f}(p) \in [a, b]\}$ is equal to $b - a$. The second property only serves to make $\overline{f}$ well-defined. The first property is what matters: this property defines the ordering of the points $\overset{i}{\times} F_i^k$ induced by $\overline{f}$. If any valid ordering $\overline{f}$ of a space-filling curve f can be defined as the visible order on F_i^k of some valid ordering $\overline{g}$ of g, then we say that *g shows f on face F_i^k*.

Using the above terminology, we get the following:

Lemma 1 *A set F of b-regular space-filling curves, containing a d-dimensional space-filling curve f_d for any $d \geq 1$, is interdimensionally consistent if and only if each curve f_d (for $d > 1$) shows f_{d-1} on each front face.*

3.3 Orientations

Recall from Section 2.3 that for each of the b^d regions $S(r)$ in a d-dimensional b-regular mono-curve, a permutation $a(r)$ defines the rotation component of the transformation that maps the curve through the unit hypercube to the curve in region $S(r)$. When we expand the recursion, we find subregions within regions. We can identify each such region by a sequence R in the following way. Let $\text{prefix}(R)$ be R without its last element, let $\text{last}(R)$ be the last element of R, and let $\langle \rangle$ be the empty sequence. Then $S(\langle \rangle)$ is the d-dimensional unit hypercube, and $S(R)$ is region $S(\text{last}(R))$ within region $S(\text{prefix}(R))$. The rotation component $a(R)$ of the transformation that maps the curve through the unit hypercube to the curve within $S(R)$, is now the permutation $a(\text{last}(R))$ applied to the permutation $a(\text{prefix}(R))$, where $a(\langle \rangle)$ is the identity permutation.

We say that a b-regular mono-curve has neutral orientation, if, in the limit for increasing recursion depth ℓ, all $d!$ permutations are equally frequent among the permutations $a(R)$ of the regions identified by sequences R of length ℓ. In other words, if R is a long random sequence of d-digit base-b numbers, then the probability that $a(R)$ is any particular permutation should be $1/d!$.

Lemma 2 *A d-dimensional b-regular mono-curve has neutral orientation if and only if there is a number ℓ, such that each of the $d!$ possible permutations of d numbers can be constructed as the composition of ℓ permutations from the set $\{a(0), ..., a((b-1)_d)\}$.*

Proof: Let all possible permutations be numbered from 1 to $d!$, where permutation 1 is the identity permutation. Let P be the $d! \times d!$ matrix defined by $P_{ij} = z/b^d$ if exactly z out of b^d regions $S(r)$ have the permutation that, applied after permutation i, results in permutation j. Note that for each permutation $a(r)$ used in z regions $S(r)$, we put an entry with value z/k^d in each row i (namely at P_{ij}, where j is the permutation that results from applying $a(r)$ to permutation i) and in each column j (namely at P_{ij}, where i is the permutation that results from applying $\overline{a}(r)$ to permutation j). Thus each row sums up to one and each column sums up to one. Let u be the $d!$-vector with $u_1 = 1$ and $u_i = 0$ for all $i > 1$.

Now, if we choose a random region identified by a sequence R of length ℓ, the probability that it has permutation i is given by element i of the vector $P^\ell u$. As ℓ goes to infinity, $P^\ell u$ converges if and only if P is a regular square matrix, that is, if and only if there is a k such that all elements of P^k are strictly positive. Note that all entries in the first column of P^k are strictly positive if and only if each permutation can be constructed from the identity permutation as the composition of k permutations from the set $\{a(0), ..., a((b-1)_d)\}$; in this case each permutation can actually be constructed from any other permutation and therefore all entries in all columns of P^k are strictly positive.

What is left to prove is that if there is a k such that all entries of P^k are strictly positive, then $\lim_{\ell \to \infty} P^\ell u$ converges to a vector of which all elements have the value $1/d!$. Note that the vector to which $\lim_{\ell \to \infty} P^\ell u$ converges is the unique vector v that satisfies $Pv = v$. Since all rows of P sum up to one, the vector of which all elements are $1/d!$ is a solution to this equation. The lemma follows. $\qquad \square$

3.4 Properties of reflected Gray codes

Recall from Section 2.5 that the reflected Gray code $b\text{RGC}^d$ is defined as follows: $b\text{RGC}^1$ is the sequence $\langle 0, 1, ..., b-1 \rangle$, and $b\text{RGC}^{d+1}$, for $d \geq 1$, consists of the concatenation of the sequences $a \oplus \text{revifodd}(a, b\text{RGC}^d)$ for $a = 0, 1, ..., b-1$ in order. As noted before, two consecutive elements $b\text{RGC}^d(\check{r})$ and $b\text{RGC}^d(r)$ differ in only one digit, and the difference in value of this digit is one. In fact, we observe that the difference is in the most significant digit in which the base-b number r differs from $\check{r}$:

Lemma 3 *Let C be a reflected Gray code with base b, indexed by base-b numbers. If $C_i(\check{r}) \neq C_i(r)$, then i has the smallest value such that $\check{r}_i \neq r_i$.*

We will now explain some observations about reflected Gray codes with base 2 and 3 that are relevant to Sections 4 and 5 of this paper.

Ternary reflected Gray codes ($b = 3$)   Recall that $b\text{RGC}^d(r)$ can be obtained from r by going through the digits of r in order from r_0 to r_{d-1}: whenever a digit r_i is encountered such that $r_i = 1$, we 'reflect' all following digits, that is, we replace r_j by $b - 1 - r_j$ for all $j > i$. Note that if $b = 3$, this operation leaves the digits 1 unaffected. Thus, $3\text{RGC}_i^d(r) = 1$ if and only if $r_i = 1$, and $3\text{RGC}_i^d(r) \in \{0, 2\}$ if and only if $r_i \in \{0, 2\}$.

Lemma 4 *(i) If r_i is even, then $\overset{i}{\times} 3\text{RGC}^d(r) = 3\text{RGC}^{d-1}(\overset{i}{\times} r)$. (ii) If k is even, $\overset{i=k}{\times} 3\text{RGC}^d = 3\text{RGC}^{d-1}$.*

Proof: Under the above mentioned step-by-step transformation that transforms r into $3\text{RGC}^d(r)$, even digits remain even and do not affect other digits. Therefore we have: if $r_i \in \{0, 2\}$ (and therefore, $3\text{RGC}_i^d(r) \in \{0, 2\}$), then $\overset{i}{\times} 3\text{RGC}^d(r) = 3\text{RGC}^{d-1}(\overset{i}{\times} r)$. This establishes part (i) of the lemma.

Furthermore, note that each prefix of $3\text{RGC}^d(r)$ only depends on the corresponding prefix of r. Therefore, selecting all pairs $(r, 3\text{RGC}^d(r))$ such that $3\text{RGC}_i^d(r) = k$, for certain values $i \in \{0, ..., d-1\}$ and $k \in \{0, 2\}$, results in selecting b^i bundles with consecutive values of r, such that the values of r in each bundle are exactly all values that have a common prefix of $i + 1$ digits, and different bundles differ in at least one of the first i digits of r. Thus, replacing each selected pair $(r, 3\text{RGC}^d(r))$ by $(\overset{i}{\times} r, \overset{i}{\times} 3\text{RGC}^d(r)) = (\overset{i}{\times} r, 3\text{RGC}^{d-1}(\overset{i}{\times} r))$ results in a sequence ordered by $\overset{i}{\times} r$, and thus, the second elements of the pairs constitute exactly the sequence 3RGC^{d-1} in order. This establishes part (ii) of the lemma. $\qquad \square$

Observe that $3\text{RGC}^d((2)_d) = (2)_d$.

Binary reflected Gray codes $(b = 2)$ Recall again that $b\text{RGC}^d(r)$ can be obtained from r by going through the digits of r in order from r_0 to r_{d-1}: whenever a digit r_i is encountered such that $r_i = 1$, we 'reflect' all following digits, that is, we replace r_j by $b - 1 - r_j$ for all $j > i$. For ease of notation, define $r_{-1} = 0$. If $b = 2$, one can verify easily by induction that the end result is the following:

Lemma 5 $2\text{RGC}_i^d(r) = |r_i - r_{i-1}|$.

This leads to the following lemma:

Lemma 6 *(i) If $r_i = r_{i-1}$ or $i = d - 1$, then $\overset{i}{\times} 2\text{RGC}^d(r) = 2\text{RGC}^{d-1}(\overset{i}{\times} r)$. If $i < d - 1$ and $r_i \neq r_{i-1}$, then $\overset{i}{\times} 2\text{RGC}^d(r) = \overset{i}{\ominus} 2\text{RGC}^{d-1}(\overset{i}{\times} r)$. (ii) For $0 \leq i < d$ we have $\overset{i=0}{\times} 2\text{RGC}^d = 2\text{RGC}^{d-1}$; for $0 \leq i < d - 1$ we have $\overset{i=1}{\times} 2\text{RGC}^d = \overset{i}{\ominus} 2\text{RGC}^{d-1}$.*

Proof: From Lemma 5 we get that the removal of a single digit r_i does not affect the remaining digits in any of the following cases: (a) $r_i = r_{i-1}$ (and thus, $2\text{RGC}_i^d(r) = 0$); and (b) $i = d - 1$ (because there are no further digits). Thus, in these cases we would also have $\overset{i}{\times} 2\text{RGC}^d(r) = 2\text{RGC}^{d-1}(\overset{i}{\times} r)$. On the other hand, if we remove a digit r_i such that $i < d - 1$ and $r_i \neq r_{i-1}$ (and thus, $2\text{RGC}_i^d(r) = 1$), the next digit of the Gray code changes. Thus we get $\overset{i}{\times} 2\text{RGC}^d(r) = \overset{i}{\ominus} 2\text{RGC}^{d-1}(\overset{i}{\times} r)$.

Following the same reasoning as above in the proof of Lemma 4, selecting all pairs $(r, 2\text{RGC}^d(r))$ such that $2\text{RGC}_i^d(r) = 0$ (for any i) or such that $2\text{RGC}_{d-1}^d(r) = 1$ and replacing each selected pair $(r, 2\text{RGC}^d(r))$ by $(\overset{i}{\times} r, \overset{i}{\times} 2\text{RGC}^d(r)) = (\overset{i}{\times} r, 2\text{RGC}^{d-1}(\overset{i}{\times} r))$ results in the sequence 2RGC^{d-1} in order. On the other hand, if we remove a digit r_i such that $2\text{RGC}_i^d(r) = 1$, the next digit of the Gray code changes. Thus, in this case we get: $\overset{i}{\times} 2\text{RGC}^d(r) = \overset{i}{\ominus} 2\text{RGC}^{d-1}(\overset{i}{\times} r)$; selecting all pairs $(r, 2\text{RGC}^d(r))$ such that $2\text{RGC}_i^d(r) = 1$ (for any $i < d - 1$) and replacing each selected pair $(r, 2\text{RGC}^d(r))$ by $(\overset{i}{\times} r, \overset{i}{\times} 2\text{RGC}^d(r)) = (\overset{i}{\times} r, \overset{i}{\ominus} 2\text{RGC}^{d-1}(\overset{i}{\times} r))$ results in the sequence $\overset{i}{\ominus} 2\text{RGC}^{d-1}$ in order. $\qquad\square$

Observe that $2\text{RGC}^d((1)_d) = 1(0)_{d-1}$.

4   Extradimensional 3-regular curves

In this section we study 3-regular vertex-continuous mono-curves, as defined in Section 2.2. We will call such curves *mono-Wunderlich curves*, after Wunderlich [17]. Besides Peano's curve, there are many other two-dimensional mono-Wunderlich curves: Figure 7 shows a selection of order-preserving mono-Wunderlich curves. Four of these curves we will generalize to higher dimensions: Peano's curve, the coil curve [8] (Luxburg's variation 1 [13]), Luxburg's variation 2 [8, 13]—which we will henceforth call *half-coil*—and the Meurthe curve [8].

4.1   Interdimensionally consistent mono-Wunderlich curves

Each curve described in this section is defined by a single recursive rule. Each curve traverses the 3^d subcubes of the unit hypercube in the order of the ternary reflected Gray code explained in Section 2.5, that is, for each region $S(r)$ we have:
$$c(r) = 3\text{RGC}_d(r).$$

In recursion, this results in each subcube being traversed from one corner to a corner that is opposite of it in all dimensions. To make sure that these curve fragments connect up properly, we define the reflections $m(r)$ as follows:
$$m_i(r) = (\text{digitsum}(r) - r_i) \bmod 2.$$

With these definitions of $c(r)$ and $m(r)$ we get a proper mono-Wunderlich curve, regardless of our choice of the permutations $a(r)$ (Lemma 7 in Section 4.3 proves this). As we will prove in Section 4.4, the following choices for $a(r)$ result in interdimensionally consistent sets of mono-Wunderlich curves:

- generalized Peano curves: $a(r)$ is simply the identity permutation ($a_i(r) = i$), regardless of r;

- generalized coil curves: $a(r)$ is the reversal permutation ($a_i(r) = d - 1 - i$), regardless of r;

- generalized half-coil curves: $a(r)$ is the identity permutation if r is odd, and the reversal permutation if r is even;
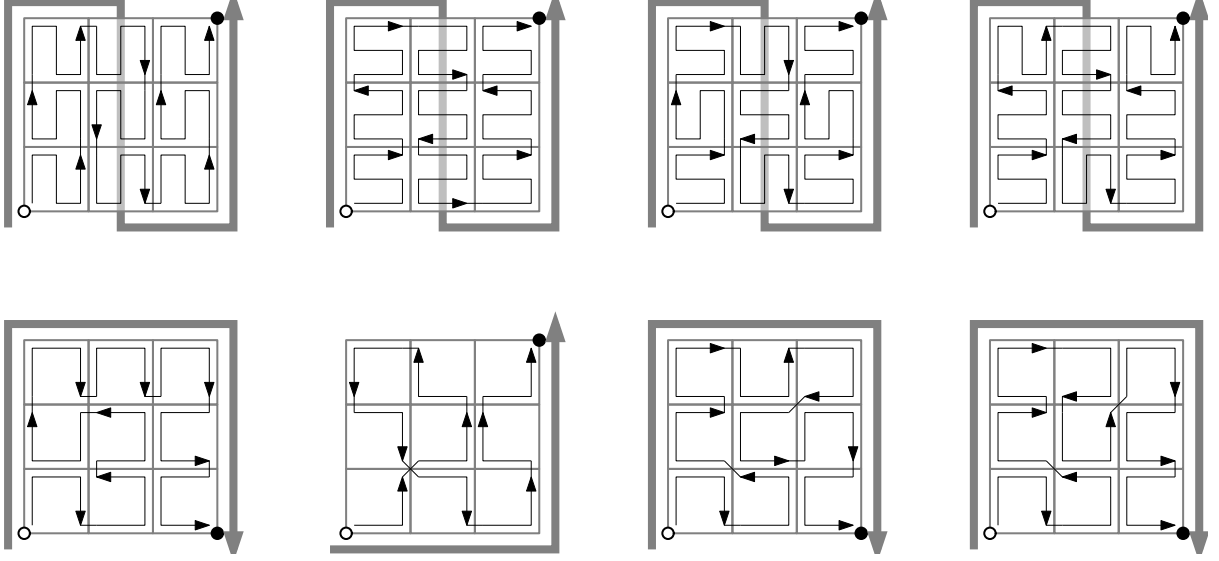
Figure 7: Some examples of order-preserving mono-Wunderlich curves. On top we see four curves that fit Wunderlich's Serpentine framework, from left to right: Peano, coil, half-coil, and Meurthe. Below, from left to right: the Я- or mirrored R-curve (Wunderlich's Meander), and three unnamed curves.

Table 1: Three-dimensional mono-Wunderlich curves from interdimensionally consistent sets

rank	loc.	permutation Pea/coil/$\frac{1}{2}$cl/Mt Mt	inv. Mt	refl.	exit		rank	loc.	permutation Pea/coil/$\frac{1}{2}$cl/Mt Mt	inv. Mt	refl.	exit
000	000	012/210/210/210	210	000	$\frac{1}{3}(1,1,1)$		112	112	012/210/210/102	102	110	$\frac{1}{3}(1,1,3)$
001	001	012/210/012/210	210	110	$\frac{1}{3}(0,0,2)$		120	102	012/210/012/120	201	011	$\frac{1}{3}(2,0,2)$
002	002	012/210/210/102	102	000	$\frac{1}{3}(1,1,3)$		121	101	012/210/210/120	201	101	$\frac{1}{3}(1,1,1)$
010	012	012/210/012/210	210	101	$\frac{1}{3}(0,2,2)$		122	100	012/210/012/012	012	011	$\frac{1}{3}(2,0,0)$
011	011	012/210/210/210	210	011	$\frac{1}{3}(1,1,1)$		200	200	012/210/210/210	210	000	$\frac{1}{3}(3,1,1)$
012	010	012/210/012/102	102	101	$\frac{1}{3}(0,2,0)$		201	201	012/210/012/210	210	110	$\frac{1}{3}(2,0,2)$
020	020	012/210/210/120	201	000	$\frac{1}{3}(1,3,1)$		202	202	012/210/210/102	102	000	$\frac{1}{3}(3,1,3)$
021	021	012/210/012/120	201	110	$\frac{1}{3}(0,2,2)$		210	212	012/210/012/210	210	101	$\frac{1}{3}(2,2,2)$
022	022	012/210/210/012	012	000	$\frac{1}{3}(1,3,3)$		211	211	012/210/210/210	210	011	$\frac{1}{3}(3,1,1)$
100	122	012/210/012/210	210	011	$\frac{1}{3}(2,2,2)$		212	210	012/210/012/102	102	101	$\frac{1}{3}(2,2,0)$
101	121	012/210/210/210	210	101	$\frac{1}{3}(1,3,1)$		220	220	012/210/210/120	201	000	$\frac{1}{3}(3,3,1)$
102	120	012/210/012/102	102	011	$\frac{1}{3}(2,2,0)$		221	221	012/210/012/120	201	110	$\frac{1}{3}(2,2,2)$
110	110	012/210/210/210	210	110	$\frac{1}{3}(1,1,1)$		222	222	012/210/210/012	012	000	$\frac{1}{3}(3,3,3)$
111	111	012/210/012/210	210	000	$\frac{1}{3}(2,2,2)$							

- generalized Meurthe curves: when $r_i \in \{0,1\}$, then $a_i(r)$ is the number of zeros and ones among $r_{i+1},...,r_{d-1}$; when $r_i = 2$, then $a_i(r)$ is $d-1$ minus the number of twos in $r_{i+1},...,r_{d-1}$. Alternatively, we can define $a(r)$ through its inverse $\bar{a}(r)$: the inverse is constructed from the identity permutation by moving all indices i such that $r_i \in \{0,1\}$ to the front and reversing their order.

Table 4.1 shows the definitions of the three-dimensional versions of each of these curves. The inverse permutations are shown explicitly only for the Meurthe curve; for the other curves, each permutation is its own inverse. For illustration, Figure 8 shows the graphical definition of the three-dimensional Meurthe curve. Of the curves described above, in three and more dimensions only the Meurthe curves have neutral orientation—we prove this in Section 4.5. The main results of Section 4 can therefore be summarized as follows:

Theorem 1 *The Peano, coil, and half-coil curves constitute interdimensionally consistent sets of order-preserving 3-regular vertex-continuous mono-curves. The Meurthe curves constitute an interdimensionally consistent set of order-preserving 3-regular vertex-continuous mono-curves with neutral orientation.*
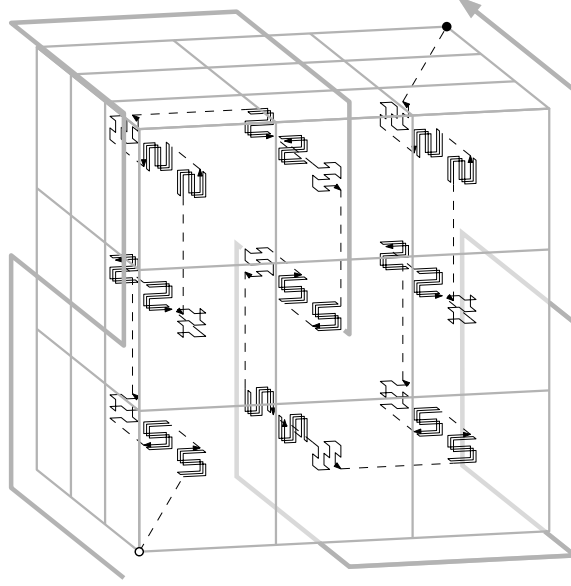
16

Figure 8: The three-dimensional Meurthe curve.

Inset 2 How the interdimensionally consistent mono-Wunderlich curves were found.

It was not straightforward to guess the permutations for the interdimensionally consistent generalizations of the mono-Wunderlich curves from the two-dimensional curves, not even for the coil and half-coil. In two dimensions there are only two numbers to permute and one cannot tell the difference between, for example, a reversal, a left rotation, and a right rotation, or the difference between the identity permutation and the permutation that reverses the whole sequence except the first element. Even in three dimensions, one cannot tell the difference between a complete reversal and a simple swap of the first and the last element.

To get started, I worked from the assumption that I could require more than what is strictly necessary: I tried to ensure that the three-dimensional curves show the two-dimensional curves on *all* faces instead of only on the front faces. Thus it was possible to deduce what the permutations in the eight corner regions had to be, and to some extent, also what the permutations in other regions had to be. Having a hypothesis for the necessary permutations in three dimensions, and with the experience gained with the harmonious Hilbert curves that are explained in Section 5, it became possible to see the patterns emerge.

Those who are familiar with our previous work in two dimensions may miss two generalizations, namely interdimensionally consistent generalizations of Wunderlich's Serpentine 011 010 110 and R-order (Wunderlich's Meander) [8]. These curves seem to be considerably more difficult to generalize in an interdimensionally consistent way than the Peano, coil, half-coil and Meurthe curves. The above approach, unfortunately, results in inconsistent requirements on the permutations for the three-dimensional generalizations of Serpentine 011 010 110 and R-order. One would therefore have to aim for a solution in which some of the back faces do *not* show the two-dimensional curve, or, for example, a mirrored version of it.

4.2 Implementation of comparison operators

In this section we discuss how to implement comparison operators for each of the four sets of curves presented above. We will first explain how to do this for points in the unit hypercube, then explain how to extend the operators to points with arbitrary non-negative coordinates, and finally how to extend the operators to points with arbitrary real coordinates. The main purpose of this section is to clarify how such comparison operators can be implemented, and to enable us to discover more properties of the curves by analysing the algorithms. For real implementations, one would have to implement the EXTRACTFIRSTDIGIT function, one would have to implement an efficient check to determine if no more digits remain to be processed, and one may consider further optimizations, such as look-up tables for permutations.

Modifications for all curves For points in the unit hypercube we can simply fill in the details of Algorithm 2.4.

To start with, we substitute "2" for "$b - 1$".

Second, we implement the reflections. The condition $m_j(rank) = 1$ translates to: $rank[j]$ is odd and $rank$ has even digit sum, or $rank[j]$ is even and $rank$ has odd digit sum. Therefore we implement the

reflections correctly if we replace the condition on Line 16 by "*rank* has odd digit sum", and replace Line 12 by:

> **if** *pDigit is odd* **then**
> | *forward* ← ¬*forward*; *reflected*[*axis*[*i*]] ← ¬*reflected*[*axis*[*i*]]

Even better, we move Line 5 to just before Line 4, and remove Line 16 altogether. The only effect of this is that in some iterations of the main loop the values of *forward* and all entries of *reflected* are 'wrong', but since all of them are wrong, the 'error' made on Line 9 will be undone on Line 10 or 11 (whichever is applicable).

A last simplification we make, is that we will work with reflected versions of *pDigit* and *qDigit* whenever *forward* is **false**. This is implemented by changing the condition in Line 8 to (*reflected*[*axis*[*i*]] = *forward*); changing the return values on Line 10 to **true** and **false**, respectively; and changing Line 11 to simply *rank*[*i*] ← *pDigit*. This may change the value of *pDigit* as used on Line 12, but it does not change the parity of *pDigit*, which is all that matters.

Thus we get Algorithm 4.1.

Algorithm 4.1: Framework for implementation of Peano, coil, half-coil and Meurthe curves

1	**for** $i \leftarrow 0$ **to** $d-1$ **do** *reflected*[*i*] ← **false**
2	**for** $i \leftarrow 0$ **to** $d-1$ **do** *axis*[*i*] ← *i* // maintains which entry in each row of T is non-zero
3	initialize an array *altAxis*[0..*d* − 1] // for a new value of *axis* under construction
4	*forward* ← **true**
5	**repeat**
6	**for** $i \leftarrow 0$ **to** $d-1$ **do**
7	*pDigit* ← EXTRACTFIRSTDIGIT(*p*[*axis*[*i*]]); *qDigit* ← EXTRACTFIRSTDIGIT(*q*[*axis*[*i*]])
8	**if** *reflected*[*axis*[*i*]] = *forward* **then**
9	*pDigit* ← (2 − *pDigit*); *qDigit* ← (2 − *qDigit*)
10	**if** *pDigit* < *qDigit* **then return true else if** *pDigit* > *qDigit* **then return false**
11	*rank*[*i*] ← *pDigit*
12	**if** *pDigit is odd* **then**
13	*forward* ← ¬*forward*; *reflected*[*axis*[*i*]] ← ¬*reflected*[*axis*[*i*]]
14	**for** $i \leftarrow 0$ **to** $d-1$ **do** *altAxis*[*i*] ← *axis*[$\overline{a}_i$(*rank*)]
15	swap *axis* and *altAxis*
16	**until** *all remaining digit strings p*[0], ..., *p*[*d* − 1] *and q*[0], ..., *q*[*d* − 1] *are empty*
17	**return false** // *p* and *q* are equal

Peano curves Since $\overline{a}_i(rank) = i$, regardless of the value of *rank*, we can simply omit Lines 2, 3, 11, 14 and 15 from Algorithm 4.1, and replace each occurrence of "*axis*[*i*]" by "*i*".

Coil curves Since $\overline{a}_i(rank) = d-1-i$, regardless of the value of *rank*, we can implement the updating of *axis* as follows. Omit Lines 11 and 14, and replace Line 3 by:

> **for** $i \leftarrow 0$ **to** $d-1$ **do** *altAxis*[*i*] ← (*d* − 1 − *i*)

Half-coil curves In half-coil curves, the permutation is reversed if and only if *rank* is even, that is, if and only if *forward* remains unchanged. We implement this as follows. Omit Lines 11 and 14. Before Line 6, add:

> *direction* ← *forward*

Replace Line 15 by:

> **if** *forward* = *direction* **then** swap *axis* and *altAxis*

Meurthe curves   Replace Line 14 by:

> $indexForNextTwo \leftarrow d$; **for** $i \leftarrow 0$ **to** $d-1$ **do if** $rank[i] = 2$ **then** decrement $indexForNextTwo$
> $indexForNextZeroOrOne \leftarrow indexForNextTwo - 1$
> **for** $i \leftarrow 0$ **to** $d-1$ **do**
> > **if** $rank[i] = 2$ **then**
> > > $altAxis[indexForNextTwo] \leftarrow axis[i]$; increment $indexForNextTwo$
> >
> > **else** $rank[i]$ is 0 or 1
> > > $altAxis[indexForNextZeroOrOne] \leftarrow axis[i]$; decrement $indexForNextZeroOrOne$

Of course the first **for** loop can be avoided by integrating it into the loop on Line 6–13 of Algorithm 4.1. When implementing this curve, make sure that the algorithm applies $\bar{a}$ (like the above piece of code does), not a. A mistake is easily made and can be hard to detect: as can be seen in Table 4.1, in the three-dimensional Meurthe curve $a(r)$ equals $\bar{a}(r)$ for most, but not all, values of r.

Points with non-negative coordinates   We can extend our implementation so that it can also compare points outside the unit hypercube. To this end, we should extend the space-filling curve outside the unit hypercube. This can be done by considering the unit hypercube as the first subregion of a larger cube. The easiest is to consider it as the first subregion of the first subregion, because for all four sets of curves, the first subregion of the first subregion is neither reflected nor rotated. Thus, the outcome of a comparison between two points that lie in the unit hypercube already, remains the same. Applying this idea recursively, we get the following algorithm:

> **while** *any coordinate of p or q is at least 1* **do** divide all coordinates of p and q by 9
> run Algorithm 4.1 on p and q

Points with negative coordinates   The above implementation does not work for points with negative coordinates. We could try to solve this by considering the unit hypercube to be the subregion in the centre of a large cube. However, in that case the entrance point of the larger cube does not lie in the origin anymore, and we loose the property that the curve traverses points in an axis-parallel hyperplane through the origin in the same order as the lower-dimensional curve. We can solve this as follows.

Note that, for the coordinates common to all points of an axis-parallel hyperplane through the origin, the implementations of the Peano, coil and half-coil curves effectively do not distinguish between the digits zero and two: in Line 9, zeros may become twos and vice versa, but as these digits remain even and remain the same between p and q, this does not affect the decisions taken by the algorithm, and they do not affect *forward* and *reflected*. This is also the case for the implementation of the Meurthe curves: a digit zero or two for coordinate i only affects where this coordinate ends up between the other coordinates in the permutation on Line 14, but it does not affect the permutation of the other coordinates relative to each other. This is all that matters, since coordinate i, being shared by p and q, is not going to decide which of p or q comes first in the order in any case. Therefore, all four sets of mono-Wunderlich curves presented in this section remain consistent even if the origin of the coordinate system is translated into the hypercube by any vector whose coordinates, in ternary representation, only contain zeros and twos.

Therefore, to allow for negative coordinates, we could consider, for $k = 0, 1, 2, ...$, each hypercube $U_k = [\frac{1}{4} - \frac{1}{4}81^k, \frac{1}{4} + \frac{3}{4}81^k)^d$ to be the last subregion of the first subregion of the last subregion of the first subregion of the hypercube $U_{k+1} = [\frac{1}{4} - \frac{1}{4}81^{k+1}, \frac{1}{4} + \frac{3}{4}81^{k+1})^d$ (note that the last of the first of the last of the first subregion is never reflected nor rotated, in all four sets of curves). Using ternary numbers, we write the coordinates of U_k as $[\frac{1}{11} - \frac{1}{11}10\,000^k, \frac{1}{11} + \frac{10}{11}10\,000^k)^d$. We will continue to use ternary numbers in the rest of this paragraph until further notice. Define $s_0(x) = x$ and let $s_1(x) = (x+202)/10\,000$ be the function that maps the coordinate range $[\frac{1}{11} - \frac{1}{11}10\,000^k, \frac{1}{11} + \frac{10}{11}10\,000^k]$ of U_k to the coordinate range $[\frac{1}{11} - \frac{1}{11}10\,000^{k-1}, \frac{1}{11} + \frac{10}{11}10\,000^{k-1}]$ of U_{k-1}. Let $s_k(x)$, for $k > 1$, be given by $s_k(x) = s_1(s_{k-1}(x))$. We can now define a local coordinate system for each hypercube U_k, were $s_k(x)$ gives the value in the local coordinate system that corresponds to x in the global coordinate system. The origin of the unit hypercube U_0 now lies at position $(s_k(0), s_k(0), ...)$ in the local coordinate system of hypercube U_k. Note that $s_k(0)$ is the number $0.(0202)_k$. Thus, the origin lies at coordinates that only contain zeros and twos in the local coordinate system of each hypercube U_k, and hence, the space-filling curve filling U_k is extradimensional to the lower-dimensional space-filling curves with respect to this origin.

The following algorithm (with numbers in decimal format again) zooms out to the smallest hypercube U_k that contains the points to be compared, while converting the coordinates to the local coordinate system of U_k:

> **while** *any coordinate of p or q is less than zero or more than one* **do**
> add 20 to all coordinates of p and q // adding the ternary number 202
> divide all coordinates of p and q by 81   // dividing by the ternary number 10 000
> run Algorithm 4.1 on p and q

Note that this solution produces different results than the solution for points with non-negative coordinates that was presented before.

4.3 Proof of vertex-continuity

Lemma 7 *The generalizations of the Peano, coil, half-coil and Meurthe curves defined above, indeed constitute mono-Wunderlich curves.*

Proof: All we have to prove is that the curves are vertex-continuous.

Recall that the curves start in the region with location $(0)_d$, and end in the region with location $(2)_d$. Thus, for the first and the last subregion r, we have $m_i(r) = 0$ for all $0 \leq i < d$. As a result, in the recursion within $S(0)$, the curve will always start in the region closest to the origin, and in the recursion within $S((2)_d)$, the curve will always end in the region farthest from the origin. Thus the entrance gate of the curve is at the origin, and the exit gate is at the corner of the d-dimensional hypercube that is farthest from the origin.

Thus, for the entrance gate of a subregion $S(r)$ to be connected to the exit gate of the previous subregion $S(\check{r})$, region $S(r)$ should be mirrored in all dimensions with respect to $S(\check{r})$, except in the one dimension in which the locations of $S(r)$ and $S(\check{r})$ differ. More precisely, (i) if $c_i(\check{r}) < c_i(r)$, we should have $m_i(\check{r}) = m_i(r) = 0$; (ii) if $c_i(\check{r}) = c_i(r)$, we should have $m_i(\check{r}) + m_i(r) = 1$, and (iii) if $c_i(\check{r}) > c_i(r)$, we should have $m_i(\check{r}) = m_i(r) = 1$.

First consider case (i): $c_i(\check{r}) < c_i(r)$. Since $c_i(\check{r}) \neq c_i(r)$, we have that i has the smallest value such that $\check{r}_i \neq r_i$, and therefore, $r_i = \check{r}_i + 1$. We have $\check{r}_0...\check{r}_{i-1} = r_0...r_{i-1}$. Moreover, since $c_i(\check{r}) < c_i(r)$, we have, by definition, $3\text{RGC}_i(\check{r}) < 3\text{RGC}_i(r)$, and the number of reflections caused by the digits $r_0, ..., r_{i-1}$, that is, the number of odd digits among $3\text{RGC}_0(r), ..., 3\text{RGC}_{i-1}(r)$, must be even. Since $3\text{RGC}_j(r)$ is odd if and only if r_j is odd, for any $j \in \{0, ..., d-1\}$, this implies that the number of odd digits among $r_0, ..., r_{i-1}$ must be even. Therefore, $\sum_{j=0}^{i-1} \check{r}_j = \sum_{j=0}^{i-1} r_j \equiv 0 \pmod 2$. By definition, $r = \check{r} + 1$, and therefore, since r and $\check{r}$ are ternary numbers, we have $\check{r}_{i+1}...\check{r}_{d-1} = (2)_{d-1-i}$ and $r_{i+1}...r_{d-1} = (0)_{d-1-i}$, so $\sum_{j=i+1}^{d-1} \check{r}_j \equiv \sum_{j=i+1}^{d-1} r_j \equiv 0 \pmod 2$. Recall that $m_i(r)$ is defined as $(\text{digitsum}(r) - r_i) \bmod 2$, so we have $m_i(\check{r}) = (\sum_{j=0}^{i-1} \check{r}_j + \sum_{j=i+1}^{d-1} \check{r}_j) \bmod 2 = 0 = (\sum_{j=0}^{i-1} r_j + \sum_{j=i+1}^{d-1} r_j) \bmod 2 = m_i(r)$, QED.

In case (ii), $c_i(\check{r}) = c_i(r)$, we know that i is *not* the smallest value j such that $r_j \neq \check{r}_j$. We have either $i < j$ and $r_i = \check{r}_i$, or $i > j$ and $\check{r}_i = 2$ and $r_i = 0$; in both cases $r_i \equiv \check{r}_i \pmod 2$. Note that for ternary numbers, $r = \check{r} + 1$ implies $\text{digitsum}(r) \equiv \text{digitsum}(\check{r}) + 1 \pmod 2$. Thus we have $m_i(r) \equiv \text{digitsum}(r) - r_i \equiv \text{digitsum}(\check{r}) + 1 - r_i \equiv \text{digitsum}(\check{r}) + 1 - \check{r}_i \equiv m_i(\check{r}) + 1 \pmod 2$, which implies $m_i(r) = (m_i(\check{r}) + 1) \bmod 2$, QED.

Third, consider case (iii): $c_i(\check{r}) > c_i(r)$. The proof is analogous to case (i), except that now, since $c_i(\check{r}) > c_i(r)$, the number of odd digits among $r_0...r_{i-1}$ must be odd, and thus we get $m_i(\check{r}) = (\sum_{j=0}^{i-1} \check{r}_j + \sum_{j=i+1}^{d-1} \check{r}_j) \bmod 2 = 1 = (\sum_{j=0}^{i-1} r_j + \sum_{j=i+1}^{d-1} r_j) \bmod 2 = m_i(r)$, QED. $\square$

4.4 Proof of interdimensional consistency

In this section we will prove that the above mentioned generalizations of the Peano, coil, half-coil and Meurthe curves are interdimensionally consistent. Arguments to this effect were also given in the paragraph of negative coordinates in Section 4.2. However, those arguments depend on the specific tie-breaking rule for points that lie on the boundary between regions, that is, the choice of $\overline{f}$, that is implemented in the pseudocode in this paper. Moreover, we need to develop a different proof technique, because in Section 5, a look at the pseudocode will not reveal the secrets of the curves as easily. Therefore, in the present section, we will prove interdimensional consistency directly from the definitions of

the curves, rather than from the implementation in pseudocode. I will first describe the part of the proof that is common to all four generalizations, and after that we will discuss the details of each separate generalization.

Consider any of the four sets of space-filling curves defined above, containing a d-dimensional space-filling curve f_d for any $d \geq 1$. As observed in Lemma 1, to prove that the set is interdimensionally consistent, it is sufficient to show that each curve f_d (for $d > 1$) shows f_{d-1} on each front face. In fact, we will prove something stronger, namely that each curve f_d shows f_{d-1} on *each* face.

Below, we write f for f_d, and by c, a, m, o, M, and τ we denote the location, permutation, reflection, translation, transformation matrix and transformation functions that define f. We write f' for f_{d-1}, and by c', a', m', o', M' and τ' we denote the location, permutation, reflection, translation, transformation matrix and transformation functions that define f'. Our proof that each curve f shows f' on each face goes by induction on increasing level of refinement. More precisely, for a level of refinement ℓ, we consider the recursive balanced subdivision of the $(d-1)$-dimensional unit hypercube U' into $3^{\ell(d-1)}$ regions, and prove the following induction hypothesis: the visible order of f on any face $F' = F_i^k$ visits these regions in the same order as f'.

Base case $\ell = 1$ As a base case, consider refinement level $\ell = 1$, and a face $F' = F_i^k$. Recall that f is defined based on subdivision of a d-dimensional unit hypercube—which we will denote U—into 3^d regions $S(0), ..., S((2)_d)$. This induces a subdivision of F' into 3^{d-1} regions $F'(0), ..., F'((2)_{d-1})$, each of which is a $(d-1)$-dimensional face of one of the regions $S(0), ..., S((2)_d)$. Let $F'(0), ..., F'((2)_{d-1})$ be indexed in the order in which the d-dimensional regions that contain them appear in $S(0), ..., S((2)_d)$. Thus, there is a monotonously increasing function $\sigma : \{0, ..., (2)_{d-1}\} \rightarrow \{0, ..., (2)_d\}$ such that $F'(r')$ is a face of $S(\sigma(r'))$. Note that for $r' \in \{0, ..., (2)_{d-1}\}$, the locations of the regions $F'(r')$ within U' are given by $\overset{i}{\times} c(\sigma(r')) = \overset{i}{\times} 3\text{RGC}^d(\sigma(r'))$. Furthermore, the co-domain of σ consists of exactly those values r such that $c_i(r) = 3\text{RGC}_i^d(r) = 2k$. Thus, the locations of the regions $F'(r')$ within U', in order of increasing r', form the sequence $\overset{i=2k}{\times} 3\text{RGC}^d$, which, by Lemma 4, is exactly the sequence $3\text{RGC}^{d-1} = c'$. Hence, the visible order of f on any face $F' = F_i^k$ visits the regions of refinement level $\ell = 1$ in the same order as f'.

Induction We will now prove that if the induction hypothesis holds for refinement level $\ell - 1$, then it also holds for refinement level ℓ. Again, consider a face $F' = F_i^k$, subdivided into 3^{d-1} top-level regions $F'(0), ..., F'((2)_{d-1})$, each subdivided into $3^{(\ell-1)(d-1)}$ smaller regions. By the analysis of the base case, the 3^{d-1} top-level regions $F'(0), ..., F'((2)_{d-1})$ of F' are traversed in the correct order. What is left to prove, is that for each $r' \in \{0, ..., (2)_{d-1}\}$, the visible order of the region $F'(r')$ traverses its recursive subdivision into $3^{(\ell-1)(d-1)}$ smaller regions in the correct order.

The transformation $\tau(r)$ that transforms the curve f through U into the section of f through $S(\sigma(r'))$, also transforms a face F of U into $F'(r')$. Denote this face F by $F^*(r')$, denote $a_i(\sigma(r'))$ by j, and observe that we have:
$$F^*(r') = F_j^{|k - m_i(\sigma(r'))|}.$$
Note that by the inductive assumption $F^*(r')$ shows the correct order f' of its subregions of level $\ell - 1$ (which map to subregions of level ℓ on F'); all that is left to prove is that the transformation $\tau(\sigma(r'))$ that operates on $S(\sigma(r'))$ transforms f' on $F^*(r')$ correctly into f' on $F'(r')$, that is, we need to show
$$\overset{i}{\times} \tau(\sigma(r'))(\overset{j=2k}{\wedge} U') = \tau'(r')(U'),$$
where $\overset{j=2k}{\wedge} U'$ is the operation that inserts a coordinate $p_j = 2k$ in the coordinate sequence of each point $p \in U'$. Recall that $\tau(r)$ is the function $p \rightarrow \frac{1}{k}M(r)p + o(r)$, so what we have to prove is:
$$\overset{i}{\times} (\frac{1}{k}M(\sigma(r'))(\overset{j=2k}{\wedge} p) + \overset{i}{\times} o(\sigma(r'))) = \frac{1}{k}M'(r')p + o'(r').$$

By the definition of $s(r')$ the translations $o(\sigma(r'))$ and $o'(r')$ are such that $F^*(r')$ is mapped to $F'(r')$, so we do not need to worry about those. Our challenge is to show that the matrices $M(\sigma(r'))$ and $M'(r')$ match, that is, $M'(r')$ must equal $M(\sigma(r'))$ with row i and column j removed. Given the definition of these matrices, we can separate this challenge into two subproblems. First, the reflections should match:
$$\overset{i}{\times} m(\sigma(r')) = m'(r'), \tag{1}$$

and second, the permutations should match:

$$\underset{\times}{\overset{i}{}} a(\sigma(r')) = a'(r'). \tag{2}$$

We start with the reflections. To prove is Equation 1. Let r be $\sigma(r')$. By definition of $r = \sigma(r')$, we have $c(r) \overset{i=2k}{=} c(r')$, and hence, $3\mathrm{RGC}_i^d(r) = 2k$ and $\underset{\times}{\overset{i}{}} 3\mathrm{RGC}^d(r) = 3\mathrm{RGC}^{d-1}(r')$. Recall that $3\mathrm{RGC}_i^d(r)$ is even if and only if r_i is even. Therefore, r_i is even, and by Lemma 4, part (i), we have:

$$r' = \underset{\times}{\overset{i}{}} r = \underset{\times}{\overset{i}{}} \sigma(r'), \tag{3}$$

and:

$$r' \equiv r = \sigma(r') \pmod 2. \tag{4}$$

By the definition of the curves, $m_j'(r') \equiv \mathrm{digitsum}(r') - r_j' \pmod 2$, and because r' is a ternary number, we have $\mathrm{digitsum}(r') \equiv r' \pmod 2$. Now we have, for $j < i$, $m_j'(r') \equiv \mathrm{digitsum}(r') - r_j' \equiv r' - r_j' \equiv \sigma(r') - \sigma_j(r') \equiv m_j(\sigma(r')) \pmod 2$, and for $j \geq i$, $m_j'(r') \equiv \mathrm{digitsum}(r') - r_j' \equiv r' - r_j' \equiv \sigma(r') - \sigma_{j+1}(r') \equiv m_{j+1}(\sigma(r')) \pmod 2$. Thus $m'(r') = \underset{\times}{\overset{i}{}} m(\sigma(r'))$, QED.

Now all that is left to prove is that the permutation components are correct, that is, we need to prove Equation 2. Below we prove this separately for each of the four sets of curves described above.

Peano curves   Since $a'(r')$ and $a(\sigma(r'))$ maintain the order of all coordinate axes regardless of r' and $\sigma(r')$, we always have $a'(r') = \underset{\times}{\overset{i}{}} a(\sigma(r'))$, QED.

Coil curves   Since $a'(r')$ and $a(\sigma(r'))$ reverse the order of all axes regardless of r' and $\sigma(r')$, we always have $a'(r') = \underset{\times}{\overset{i}{}} a(\sigma(r'))$, QED.

Half-coil curves   The permutation $a'(r')$ reverses the order of all axes if r is odd, and maintains the order of all axes if r' is even. The permutation $a(\sigma(r'))$ reverses the order of all axes if $\sigma(r')$ is odd, and maintains the order of all axes if $\sigma(r')$ is even. By Equation 4 we have that r' is odd if and only if $\sigma(r')$ is odd, and thus, the permutations $a'(r')$ and $a(\sigma(r'))$ match, that is, $a'(r') = \underset{\times}{\overset{i}{}} a(\sigma(r'))$, QED.

Meurthe curves   We will again use Equation 3. Recall that the respective inverses of $a'(r')$ and $a(r)$ (where $r = \sigma(r')$) can be constructed from the identity permutation by moving all indices h such that r_h', or r_h, respectively, is 0 or 1 to the front, and reversing the order of these indices. Obviously, inserting a digit in r' at position i to obtain $r = \sigma(r')$, moving the index i in $\bar{a}(r)$, and subsequently removing index i from $\bar{a}(r)$ (which is what $\underset{\times}{\overset{i}{}} a(r)$ does), does not affect the order of the remaining indices relative to each other. Thus we have $a'(r') = \underset{\times}{\overset{i}{}} a(\sigma(r'))$, QED.

4.5   Orientation of the interdimensionally consistent mono-Wunderlich curves

By Lemma 2, a d-dimensional mono-Wunderlich curve has neutral orientation if and only if there is a number ℓ, such that each of the $d!$ possible permutations of d numbers can be constructed as the composition of ℓ permutations from a. Trivially, each 1-dimensional curve has neutral orientation. Below we analyse whether the d-dimensional curves for $d \geq 2$ have neutral orientation.

Peano curves   In the Peano curves, a only contains the identity permutation; therefore no other permutation can be constructed and the Peano curves do *not* have neutral orientation.

Coil curves   In the coil curves, a only contains the reversal permutation. As a result, there is no ℓ such that each permutation can be constructed as the composition of ℓ permutations from a, not even if $d = 2$: if ℓ is odd, only the reversal permutation can be constructed, and if ℓ is even, only the identity permutation can be constructed. Therefore the coil curves do *not* have neutral orientation.

Half-coil curves   In the half-coil curves, a only contains the identity and the reversal permutation. As a result, there is no ℓ such that each permutation can be constructed as the composition of ℓ permutations from a, unless $d = 2$. Thus, the two-dimensional half-coil curve has neutral orientation, but the higher-dimensional half-coil curves have not.

Meurthe curves In the Meurthe curves, $a(00(2)_{d-2})$ is the permutation that swaps the first two elements, and $a((2)_{d-1}0)$ is the permutation that rotates the whole sequence one step to the right. Thus, we can swap any two adjacent elements by composing at most $d + 1$ permutations: d rotations $a((2)_{d-1}0)$ and one swap $a(00(2)_{d-2})$, placed between the rotations at the point where the two elements to be swapped have been rotated to the first two positions. Since any permutation can be created from any other by at most $d(d-1)/2$ swaps of adjacent elements, we get that any permutation can be created from any other by composing at most $(d+1)d(d-1)/2$ permutations $a(00(2)_{d-2})$ and $a((2)_{d-1}0)$. However, to satisfy the condition in Lemma 2, there should be a depth ℓ such that each permutation can be created from the identity permutation by composing *exactly* ℓ permutations. In the Meurthe curves, this can indeed be done for $\ell = (d + 1)d(d - 1)/2$, since any shorter sequence of permutations can be completed by the identity permutation found at $a((2)_d)$. If follows that the Meurthe curves have neutral orientation.

Note that the analysis given above is by no means tight: in reality much less than $(d+1)d(d-1)/2$ permutations from a may suffice to produce all possible permutations. For example, in four dimensions two permutations suffice, rather than $(d + 1)d(d - 1)/2 = 30$.

5 Extradimensional 2-regular curves

In this section we study 2-regular vertex-continuous mono-curves, as defined in Section 2.2. We will call such curves *mono-Hilbert curves* [6]. In two dimensions, the properties that define a mono-Hilbert curve form a minimal sufficient set of properties that defines Hilbert's original curve (see Figure 6). Therefore one can argue, although it is ultimately a matter of taste, that in higher dimensions these properties are necessary and sufficient for a curve to be considered a true generalization of a Hilbert curve [6]. Below we will discuss two sets of mono-Hilbert curves in arbitrary dimensions: the generalization implemented by Moore [14] (citing Butz [3]), and a new generalization called *harmonious Hilbert curves*.

5.1 Standard Hilbert curves

We will first describe a class of Hilbert curves in higher dimensions, in which the order of the subregions and the connection points between them follow a very regular pattern. We call these curves *standard Hilbert curves* (by lack of a better name).

Each standard Hilbert curve is defined by a single recursive rule. Each curve traverses the 2^d subcubes of the unit hypercube in the order of the binary reflected Gray code explained in Section 2.5, that is, for each region $S(r)$ we have:
$$c(r) = 2\mathrm{RGC}_d(r).$$
Standard Hilbert curves differ in their permutation functions a, but $\bar{a}_0$ is fixed, as follows (recall that we write $\check{r}$ for $r - 1$, and we write $\hat{(r)}$ for $r + 1$):

$$
\begin{aligned}
&a_{d-1}(r) = 0 && \text{if } r = 0 \text{ (the first subregion) or } r = (1)_d \text{ (the last subregion);} \\
&a_i(r) = 0 && \text{if } 0 < r < (1)_d, \ r \text{ is odd and } c_i(r) \neq c_i(\hat{r}); \\
&a_i(r) = 0 && \text{if } 0 < r < (1)_d, \ r \text{ is even and } c_i(r) \neq c_i(\check{r}).
\end{aligned}
$$

Reflections are defined as follows:

$$
\begin{aligned}
&m_i(r) = c_i(r) && \text{if } r = 0; \\
&m_i(r) = 1 - c_i(r) && \text{if } r > 0 \text{ and } i = d - 1; \\
&m_i(r) = c_i(\check{r}) && \text{if } r > 0 \text{ and } i < d - 1.
\end{aligned}
$$

We consider two sets of standard Hilbert curves:

- Butz-Moore curves, as implemented by Moore [14], based on the work of Butz [3]: here $\bar{a}_j(r) = (\bar{a}_0(r) + j) \bmod d$.

- our new *harmonious Hilbert curves*: when $r_i \neq r_{d-1}$, then $a_i(r)$ is the number of digits among $r_{i+1}, ..., r_{d-1}$ that differ from r_{d-1}; when $r_i = r_{d-1}$, then $a_i(r)$ is $d - 1$ minus the number of digits among $r_0, ..., r_{i-1}$ that equal r_{d-1}. Alternatively, we can define $a(r)$ through its inverse $\bar{a}(r)$: the inverse is constructed from the identity permutation by moving all indices i such that $r_i \neq r_{d-1}$ to the front and reversing their order, and reversing the order of the remaining indices as well.
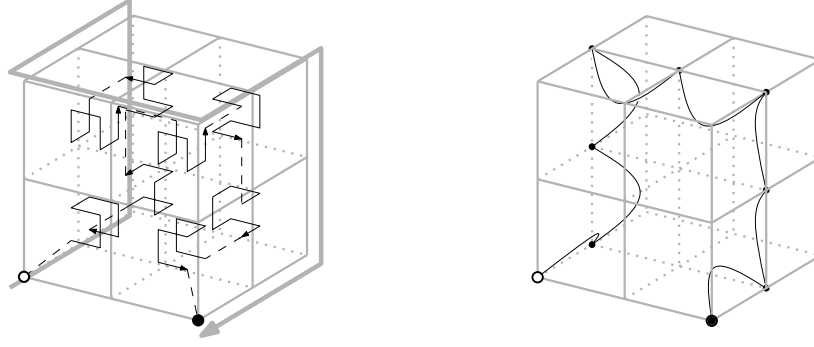
Figure 9: The three-dimensional harmonious Hilbert curve. Left: definition. Right: location of the gates between the subregions.

Table 2: Butz-Moore curves and harmonious Hilbert curves in five dimensions.

rank	loc.	permutat. B-M / HH	inv. perm. B-M / HH	refl.	exit	rank	loc.	permutat. B-M / HH	inv. perm. B-M / HH	refl.	exit
00000	00000	12340/43210	40123/43210	00000	$\frac{1}{2}(0,0,0,0,1)$	10000	11000	01234/04321	01234/04321	01001	$\frac{1}{2}(2,2,0,0,1)$
00001	00001	23401/32104	34012/32104	00000	$\frac{1}{2}(0,0,0,1,1)$	10001	11001	23401/42103	34012/32140	11000	$\frac{1}{2}(2,2,0,1,1)$
00010	00011	23401/43201	34012/34210	00000	$\frac{1}{2}(0,0,0,2,1)$	10010	11011	23401/14302	34012/30421	11000	$\frac{1}{2}(2,2,0,2,1)$
00011	00010	34012/21043	23401/21043	00011	$\frac{1}{2}(0,0,1,2,1)$	10011	11010	34012/41032	23401/21430	11011	$\frac{1}{2}(2,2,1,2,1)$
00100	00110	34012/43021	23401/24310	00011	$\frac{1}{2}(0,0,2,2,1)$	10100	11110	34012/14032	23401/20431	11011	$\frac{1}{2}(2,2,2,2,1)$
00101	00111	23401/21403	34012/31042	00110	$\frac{1}{2}(0,0,2,1,1)$	10101	11111	23401/41302	34012/31420	11110	$\frac{1}{2}(2,2,2,1,1)$
00110	00101	23401/43102	34012/32410	00110	$\frac{1}{2}(0,0,2,0,1)$	10110	11101	23401/24103	34012/32041	11110	$\frac{1}{2}(2,2,2,0,1)$
00111	00100	40123/10432	12340/10432	00101	$\frac{1}{2}(0,1,2,0,1)$	10111	11100	40123/40321	12340/14320	11101	$\frac{1}{2}(2,1,2,0,1)$
01000	01100	40123/40321	12340/14320	00101	$\frac{1}{2}(0,2,2,0,1)$	11000	10100	40123/10432	12340/10432	11101	$\frac{1}{2}(2,0,2,0,1)$
01001	01101	23401/24103	34012/32041	01100	$\frac{1}{2}(0,2,2,1,1)$	11001	10101	23401/43102	34012/32410	10100	$\frac{1}{2}(2,0,2,1,1)$
01010	01111	23401/41302	34012/31420	01100	$\frac{1}{2}(0,2,2,2,1)$	11010	10111	23401/21403	34012/31042	10100	$\frac{1}{2}(2,0,2,2,1)$
01011	01110	34012/14032	23401/20431	01111	$\frac{1}{2}(0,2,1,2,1)$	11011	10110	34012/43021	23401/24310	10111	$\frac{1}{2}(2,0,1,2,1)$
01100	01010	34012/41032	23401/21430	01111	$\frac{1}{2}(0,2,0,2,1)$	11100	10010	34012/21043	23401/21043	10111	$\frac{1}{2}(2,0,0,2,1)$
01101	01011	23401/14302	34012/30421	01010	$\frac{1}{2}(0,2,0,1,1)$	11101	10011	23401/43201	34012/34210	10010	$\frac{1}{2}(2,0,0,1,1)$
01110	01001	23401/42103	34012/32140	01010	$\frac{1}{2}(0,2,0,0,1)$	11110	10001	23401/32104	34012/32104	10010	$\frac{1}{2}(2,0,0,0,1)$
01111	01000	01234/04321	01234/04321	01001	$\frac{1}{2}(1,2,0,0,1)$	11111	10000	12340/43210	40123/43210	10001	$\frac{1}{2}(2,0,0,0,0)$

Table 5.1 shows the definitions of the five-dimensional versions of each of these curves. For illustration, Figure 9 shows the graphical definition of the three-dimensional harmonious Hilbert curve. The harmonious Hilbert curves are interdimensionally consistent and have neutral orientation (as we will see below), the Butz-Moore curves are not interdimensionally consistent (as observed in Section 1) and do not have neutral orientation. The main result of Section 5 can be summarized as follows:

Theorem 2 *The harmonious Hilbert curves constitute an interdimensionally consistent set of order-preserving 2-regular vertex-continuous mono-curves with neutral orientation.*

5.2   Implementation of comparison operators

In this section we discuss how to implement comparison operators for each of the two sets of curves presented above. We will first explain how to do this for points in the unit hypercube, then explain how to extend the operators to points with arbitrary non-negative coordinates, and finally how to extend the operators to points with arbitrary real coordinates. The main purpose of this section is to clarify how such comparison operators can be implemented, and to enable us to discover more properties of the curves by analysing the algorithms. For real implementations, one would have to implement the EXTRACTFIRSTDIGIT function, and one may consider optimizations such as look-up tables for permutations. You may find that many steps of the algorithm can be implemented and quite easily and efficiently by bit operations such as bit shifting and exclusive or. Readers who are not interested in implementations may want to skip to Section 5.3 right away.

The first non-trivial harmonious Hilbert curve is the three-dimensional curve. This curve was discovered when generating all 10,694,807 different three-dimensional mono-Hilbert curves, including curves that are not order-preserving [6]. In fact, it turned out that in three dimensions, the harmonious Hilbert curve, showing the two-dimensional Hilbert curve on five faces, is the only mono-Hilbert curve that is extradimensional to the two-dimensional Hilbert curve. There is no three-dimensional mono-Hilbert curve that shows the two-dimensional Hilbert curve on all six faces (even when allowing reflections on some faces).

Having discovered the three-dimensional curve, the question arose whether similar curves would exist in higher dimensions, and if so, how many of them. Unfortunately, the approach to finding interdimensionally consistent generalizations of mono-Wunderlich curves, as explained in Inset 2, could not be applied to mono-Hilbert curves: it would already fail in three dimensions. Instead, I programmed an exhaustive search algorithm that, given a d-dimensional curve f_d, would search all possible symmetric $(d+1)$-dimensional curves that traverse the subregions in binary reflected Gray code order, reporting those curves that are extradimensional to f_d. Using effective pruning, unique solutions for four-, five- and six-dimensional curves were quickly found. One may regard this as a somewhat independent confirmation that the proofs given in Section 5.4 are likely to be correct.

It became clear that the challenge was not to run the exhaustive search, but to understand the structure in the solutions. Freek van Walderveen was the first to recognize structure in the permutation patterns, and managed to write an efficient algorithm that correctly predicted the permutations of the seven-dimensional curve, which was subsequently verified by my exhaustive search algorithm. Insights that led to a simpler algorithm began to come when I drew a graph of the subregions whose permutations differ by only one inversion. After drawing the graph for the five-dimensional curve on a geometric drawing of the five-dimensional hypercube, it became clear that inversions in the permutations of subregions $S(r)$ correspond to inversions of adjacent digits with different values in the binary representations of the ranks r. With an analysis of the necessary base cases, this led to the simple definition given in Section 5.1, which specifies the permutation for any given rank.

Modifications for all curves For points in the unit hypercube we can simply fill in the details of Algorithm 2.4. To start with, we substitute "1" for "$b-1$".

Second, we implement the reflections. First, consider the reflections of regions with even rank r. Since r is even, we either have $r = 0$, or there is some i such that $\check{r}_{i-1}\check{r}_i...\check{r}_{d-1} = 0(1)_{d-i}$ and $r_{i-1}r_i...r_{d-1} = 1(0)_{d-i}$, while the first $i-1$ digits of $\check{r}$ and r are equal. By Lemma 5, i is now the largest i such that $c_i(r) = 1$. Furthermore, by Lemma 3, we have $c_{i-1}(r) \neq c_{i-1}(\check{r})$, and thus, by definition, $\bar{a}_0(r) = i - 1$.

If r is odd, we either have $r = (1)_d$, or there is some i such that $r_{i-1}r_i...r_{d-1} = 0(1)_{d-i}$ and $\hat{r}_{i-1}\hat{r}_i...\hat{r}_{d-1} = 1(0)_{d-i}$, and we find that i is the largest i such that $c_i(r) = 1$, $c_{i-1}(r) \neq c_{i-1}(\hat{r})$, and $\bar{a}_0(r) = i - 1$. Finally, observe that r is odd whenever the number of ones in $c(r)$ is odd, since $c(0) = 0$ and, as r increases to $(1)_d$, the value of r and the number of ones in $c(r)$ both change by one in every step.

This leads to the following simple method to determine the reflections $m(r)$ based on $c(r)$, for $0 < r < (1)_d$. Go through the digits $c_i(r)$ in order of increasing i: if $c_i(r) = 0$, set $m_i(r) = 0$; if $c_i(r) = 1$, set $m_i(r) = 1$ and set $\bar{a}_0(r) = i - 1 \pmod d$. This procedure computes $m_i(r)$ correctly except when $c_i(r) \neq c_i(\check{r})$ or $i < d-1$. Therefore we correct $m(r)$ by changing $m_{d-1}(r)$ and, when the number of ones in $c(r)$ is even, changing $m_i(r)$ for $i = \bar{a}_0(r)$. For $r = 0$, the exact same procedure also works correctly if $\bar{a}_0(r)$ is initialized to $d - 1$, as $\bar{a}_0(r)$ will remain unchanged, $m_i(r)$ will be set to zero correctly for all i, and in the end, the two corrections (of $m_{d-1}(r)$ and of $m_i(r)$ for $i = \bar{a}_0(r) = d - 1$) cancel each other's effect. For $r = (1)_d$, we have $c(r) = 1(0)_{d-1}$, and the procedure will also set $\bar{a}_0(r)$ and m_r correctly.

Therefore we implement the reflections correctly if we modify Algorithm 2.4 as follows. After Line 5, add:

$iMinusOne \leftarrow d - 1$; $altAxis[0] \leftarrow axis[iMinusOne]$

Replace Line 12 by:

if $pDigit = 1$ **then**
  $\quad forward \leftarrow \neg forward$; $reflected[axis[i]] \leftarrow \neg reflected[axis[i]]$
  $\quad altAxis[0] \leftarrow axis[iMinusOne]$
$iMinusOne \leftarrow i$

Remove Line 16. Instead, add the following before Line 17 (outside the **for** loop):

> $reflected[axis[d-1]] \leftarrow \neg reflected[axis[d-1]]$
> **if** $forward$ **then**
> > // $pDigit$ was one an even number of times
> > $reflected[altAxis[0]] \leftarrow \neg reflected[altAxis[0]]$

Even better, we move Line 5 to just before Line 4, and 'correct' $reflected[altAxis[0]]$ unconditionally. The only effect of this is that we will enter the next iteration of the **repeat**...**until** loop with wrong values for $forward$ and $reflected[axis[0]]$, but the reader may verify that this does not change the return values and $rank[i]$ as computed in the first iteration of the large **for** loop. In contrast, the value of $pDigit$ will change, and thus the values of $forward$ and $reflected[axis[0]]$ will be correct again by the end of the first iteration of the **for** loop. Thus we get Algorithm 5.1.

Algorithm 5.1: Framework for implementation of Butz-Moore and harmonious Hilbert curves

1 **for** $i \leftarrow 0$ **to** $d-1$ **do** $reflected[i] \leftarrow$ **false**
2 **for** $i \leftarrow 0$ **to** $d-1$ **do** $axis[i] \leftarrow i$ // maintains which entry in each row of T is non-zero
3 initialize an array $altAxis[0..d-1]$ // for a new value of $axis$ under construction
4 $forward \leftarrow$ **true**
5 **repeat**
6 $iMinusOne \leftarrow d-1$; $altAxis[0] \leftarrow axis[iMinusOne]$
7   **for** $i \leftarrow 0$ **to** $d-1$ **do**
8 $pDigit \leftarrow$ EXTRACTFIRSTDIGIT$(p[axis[i]])$; $qDigit \leftarrow$ EXTRACTFIRSTDIGIT$(q[axis[i]])$
9   **if** $reflected[axis[i]]$ **then**
10 $pDigit \leftarrow (1 - pDigit)$; $qDigit \leftarrow (1 - qDigit)$
11   **if** $pDigit < qDigit$ **then return** $forward$ **else if** $pDigit > qDigit$ **then return** $\neg forward$
12   **if** $forward$ **then** $rank[i] \leftarrow pDigit$ **else** $rank[i] \leftarrow (1 - pDigit)$
13   **if** $pDigit = 1$ **then**
14 $forward \leftarrow \neg forward$; $reflected[axis[i]] \leftarrow \neg reflected[axis[i]]$
15 $altAxis[0] \leftarrow axis[iMinusOne]$
16 $iMinusOne \leftarrow i$
17   **for** $i \leftarrow 1$ **to** $d-1$ **do** $altAxis[i] \leftarrow axis[\overline{a}_i(rank)]$
18 $reflected[axis[d-1]] \leftarrow \neg reflected[axis[d-1]]$; $reflected[altAxis[0]] \leftarrow \neg reflected[altAxis[0]]$
19   swap $axis$ and $altAxis$
20 **until** all remaining digit strings $p[0], ..., p[d-1]$ and $q[0], ..., q[d-1]$ are empty
21 **return false**   // p and q are equal

Butz-Moore curves   The permutations of Butz-Moore curves are always rotations (in the permutation sense of the word), therefore they can be maintained by only keeping track of $axis[0]$, without storing $axis[1..d-1]$. Thus we can simplify Algorithm 5.1 substantially and get Algorithm 5.2: here we maintain $axis[0]$ in a variable called $rotation$. We define the value of $-1 \bmod d$ as $d-1$.

Harmonious Hilbert curves   To implement the harmonious Hilbert curves, remove Lines 6, 15 and 16, and replace Line 17 by Algorithm 5.3. The first **for** loop of Algorithm 5.3 can be avoided by integrating it into the loop on Line 7–16 of Algorithm 5.1.

Points with non-negative coordinates   To enable our implementation to compare points outside the unit hypercube, we consider the unit hypercube as the first subregion of a larger cube. Again, as in Section 4.2, for the harmonious Hilbert curves it is easiest to consider the unit hypercube as the first subregion of the first subregion of a larger cube, because the first subregion of the first subregion is neither rotated nor reflected. Thus we get the following algorithm:

> **while** any coordinate of p or q is at least 1 **do** divide all coordinates of p and q by 4
> run Algorithm 5.1 for harmonious Hilbert curves on p and q

Algorithm 5.2: Implementation of the Butz-Moore curves

---

1 $forward \leftarrow$ **true**; $rotation \leftarrow 0$; **for** $i \leftarrow 0$ **to** $d-1$ **do** $reflected[i] \leftarrow$ **false**

2 **repeat**

3 $\quad i \leftarrow rotation$; $iMinusOne \leftarrow (i-1) \bmod d$; $newRotation \leftarrow iMinusOne$

4 $\quad$ **repeat**

5 $\quad\quad pDigit \leftarrow$ EXTRACTFIRSTDIGIT$(p[i])$; $qDigit \leftarrow$ EXTRACTFIRSTDIGIT$(q[i])$

6 $\quad\quad$ **if** $reflected[i]$ **then**

7 $\quad\quad\quad pDigit \leftarrow (1 - pDigit)$; $qDigit \leftarrow (1 - qDigit)$

8 $\quad\quad$ **if** $pDigit < qDigit$ **then return** $forward$; **else if** $pDigit > qDigit$ **then return** $\neg forward$

9 $\quad\quad$ **if** $pDigit = 1$ **then**

10 $\quad\quad\quad forward \leftarrow \neg forward$; $reflected[i] \leftarrow \neg reflected[i]$; $newRotation \leftarrow iMinusOne$

11 $\quad\quad iMinusOne \leftarrow i$; $i \leftarrow (i+1) \bmod d$

12 $\quad$ **until** $i = rotation$

13 $\quad reflected[iMinusOne] \leftarrow \neg reflected[iMinusOne]$

14 $\quad reflected[newRotation] \leftarrow \neg reflected[newRotation]$

15 $\quad rotation \leftarrow newRotation$

16 **until** *all remaining digit strings* $p[0], ..., p[d-1]$ *and* $q[0], ..., q[d-1]$ *are empty*

17 **return false**

Algorithm 5.3: Algorithm to compute the inverse permutation of region *rank* of a harmonious Hilbert curve, and to apply it to the permutation *axis* to obtain *altAxis*.

---

1 $numberOfOnes \leftarrow 0$; **for** $i \leftarrow 0$ **to** $d-1$ **do** $numberOfOnes \leftarrow numberOfOnes + rank[i]$

2 **if** $rank[d-1] = 0$ **then**

3 $\quad indexForNext[1] \leftarrow 0$; $indexForNext[0] \leftarrow numberOfOnes$

4 **else** $rank[d-1] = 1$

5 $\quad indexForNext[0] \leftarrow 0$; $indexForNext[1] \leftarrow d - numberOfOnes$

6 **for** $i \leftarrow d-1$ **down to** 0 **do**

7 $\quad altAxis[indexForNext[rank[i]]] \leftarrow axis[i]$; increment $indexForNext[rank[i]]$

Points with negative coordinates   One could try to extend the implementation to negative coordinates in a way similar to what is explained in Section 4.2. Recall that the basic idea is to move the origin of the coordinate system for the original, hypercube-filling curves, to somewhere properly inside the unit hypercubes. By 'zooming out', one can then obtain curves that fill the full d-dimensional space, rather than covering only points with non-negative coordinates. However, the resulting curves would only be interdimensionally consistent, if the original hypercube-filling curves retain interdimensional consistency with the redefined origin. Since I have not found any location for the origin that meets this requirement, I omit the details for extensions to negative coordinates.

5.3   Proof of vertex-continuity

In this section we will prove that the Butz-Moore and the harmonious Hilbert curves are indeed curves, that is, they are vertex-continuous. We will do so in two steps: first we prove that standard Hilbert curves are vertex-continuous, and then we proof that the Butz-Moore and the harmonious Hilbert curves are standard Hilbert curves.

Lemma 8 *Standard Hilbert curves constitute mono-Hilbert curves.*

Proof: All we have to prove is that the curves are vertex-continuous.

We first establish the location of the entrance and exit gates of the curves. Recall that the curves start in the region $S(0)$ with location $(0)_d$, and by definition $m_i(0) = 0$ for all $0 \leq i < d$. As a result, in the recursion within $S(0)$, the curve will always start in the region closest to the origin. Thus the entrance gate of the curve is at the origin. Let r be $2^d - 1 = (1)_d$. The curves end in the region $S(r)$ with

location $c(r) = 1(0)_{d-1}$, that is, the region touching the point $(1, 0, ..., 0)$. Note that $c(\check{r}) = 1(0)_{d-2}1$, and therefore, $m(r) = 1(0)_{d-2}1$. The permutation $a(r)$ is the reversal permutation. The coordinates of the last subregion within $S(r)$ are obtained by applying the permutation specified by $a(r)$ and the reflections specified by $m(r)$ to the coordinates $c(r)$: we find that the last subregion visited within $S(r)$ is again the region closest to the the point $(1, 0, ..., 0)$. Thus, in recursion, the curve will always end in the region touching the point $(1, 0, ..., 0)$, which is therefore the exit gate of the curve.

I claim that for $0 < r < 2^d$, the connecting gate between subregion $S(\check{r})$ and $S(r)$ is at the point $x(r) = (x_0(r), ..., x_{d-1}(r))$, where $x_i(r) = \frac{1}{2}(c_i(\check{r}) + c_i(r))$ for $0 \le i < d-1$, and $x_{d-1}(r) = \frac{1}{2}$. For ease of notation, we let $x(0) = (0, 0, .., 0)$ and $x(2^d) = (1, 0, ..., 0)$ be the entrance and the exit gate, respectively, of the complete curve. (Figure 9 shows an example in three dimensions). To prove that my claim is correct, we should establish that the curve within each region $S(r)$ starts in $x(r)$ and ends in $x(\hat{r})$. I leave it to the reader to verify this for the special cases of the first subregion ($r = 0$) and the last subregion ($r = 2^d - 1$). We will now see how to verify that the curve within $S(r)$ starts in $x(r)$ and ends in $x(\hat{r})$ for $0 < r < 2^d - 1$.

Observe that the location of $S(r)$ differs from the location of one of its neighbours in the ordering in coordinate c_{d-1}, and it differs from the location of the other neighbour in coordinate c_j where $j = \bar{a}_0(r)$. Thus, $j = \bar{a}_0(r)$ is the unique $j < d-1$ such that $c_j(\check{r}) \neq c_j(\hat{r})$. The entrance gate of subregion $S(r)$ is at $\tau(r)((0, 0, ..., 0))$, which is the point $p = (p_0, ..., p_{d-1})$ such that $p_i = \frac{1}{2}(c_i(r) + m_i(r))$ for $0 \le i < d$; the exit gate is at $\tau(r)((1, 0, ..., 0))$, which is the point $q = (q_0, ..., q_{d-1})$ such that $q_j = \frac{1}{2}(c_j(r) + 1 - m_j(r))$ for $j = \bar{a}_0(r)$, and $q_i = \frac{1}{2}(c_i(r) + m_i(r)) = p_i$ for all $i \in \{0, ..., d-1\} \setminus \{\bar{a}_0(r)\}$. We have to show (i) $p_i = x_i(r)$ and (ii) $q_i = x_i(\hat{r})$ for all $i \in \{0, ..., d-1\}$. For $i < d-1$, we can rewrite (i) as: $c_i(r) + m_i(r) = c_i(\check{r}) + c_i(r)$, and therefore: $m_i(r) = c_i(\check{r})$. This corresponds directly to the definition of $m_i(r)$. For $i < d-1$ and $i \neq j$, we can rewrite (ii) as: $c_i(r) + m_i(r) = c_i(r) + c_i(\hat{r}) = c_i(r) + c_i(\check{r})$; again, this corresponds directly to the definition of $m_i(r)$. For $j = \bar{a}_0(r)$, we rewrite (ii) as: $c_j(r) + 1 - m_j(r) = c_j(r) + c_j(\hat{r}) = c_j(r) + 1 - c_j(\check{r})$, and therefore: $m_j(r) = c_j(\check{r})$. Again, this corresponds directly to the definition of $m_i(r)$. Finally, for $i = d-1$, we rewrite both (i) and (ii) as: $c_i(r) + m_i(r) = 1$. This, too, corresponds directly to the definition of $m_i(r)$, QED. □

Lemma 9 *The Butz-Moore curves and the harmonious Hilbert curves defined above are standard Hilbert curves.*

Proof: What we have to prove is that the permutations a in the definition of the Butz-Moore curves and the harmonious Hilbert curves satisfy the definition of $\bar{a}_0$ in standard Hilbert curves.

For the Butz-Moore curves this is trivial, as their permutations are defined in terms of a given $\bar{a}_0$.

For the harmonious Hilbert curves the proof is more involved. First, when $r = 0$ or $r = (1)_d$, all bits of r are equal. Thus, by the definition of harmonious Hilbert curves, $a_{d-1}(r) = (d-1) - (d-1) = 0$, which is indeed consistent with the definition of standard Hilbert curves. Second, consider the case when $0 < r < 2^d - 1$ and r is odd, and let i have the lowest value such that $r_i \neq \hat{r}_i$. Then $i < d-1$, $r_i = 0$, and $r_{i+1}, ..., r_{d-1} = 1$. By the definition of harmonious Hilbert curves, we will have $a_i(r) = 0$, and by the definition of standard Hilbert curves, we will have $c_i(r) \neq c_i(\hat{r})$, and thus, indeed, $a_i(r) = 0$. Third, consider the case when $0 < r < 2^d - 1$ and r is even, and let i have the lowest value such that $r_i \neq \check{r}_i$. Then $i < d-1$, $r_i = 1$, and $r_{i+1}, ..., r_{d-1} = 0$. By the definition of harmonious Hilbert curves, we will have $a_i(r) = 0$, and by the definition of standard Hilbert curves, we will have $c_i(r) \neq c_i(\check{r})$, and thus, indeed, $a_i(r) = 0$. □

Corollary 1 *The Butz-Moore curves and the harmonious Hilbert curves defined above are mono-Hilbert curves.*

5.4   Proof of interdimensional consistency

In this section we will prove that the harmonious Hilbert curves are interdimensionally consistent. Let f_d denote the d-dimensional harmonious Hilbert curve. As observed in Section 3.2, to prove that the harmonious Hilbert curves are interdimensionally consistent, it suffices to show that each curve f_d (for $d > 1$) shows f_{d-1} on each front face. In fact, we will prove something stronger, namely that each curve f_d shows f_{d-1} on each front face, and each back face F_i^1 with $0 \le i < d-1$ shows f_{d-1} mirrored in dimension i (note that this dimension corresponds to dimension $i + 1$ in the d-dimensional space). This leaves F_{d-1}^1 as the only face for which we do not prove anything about the visible order.

Below, we write f for f_d, and by c, a, m, o, M, and τ we denote the location, permutation, reflection, translation, transformation matrix and transformation functions that define f. We write f' for f_{d-1}, and by c', a', m', o', M' and τ' we denote the location, permutation and reflection, translation, transformation matrix and transformation functions that define f'. The proof that each curve f shows f' (sometimes mirrored, as specified above) on all faces except F_{d-1}^1 goes by induction on increasing level of refinement. More precisely, for a level of refinement ℓ, we consider the recursive balanced subdivision of the $(d-1)$-dimensional unit hypercube U' into $2^{\ell(d-1)}$ regions, and prove the following induction hypothesis:

(i) the visible order of f on any face $F' = F_i^0$ with $0 \le i \le d-1$ visits these regions in the same order as f';

(ii) the visible order of f on any face $F' = F_i^1$ with $0 \le i < d-1$ visits these regions in the same order as f' mirrored in dimension i.

The proof follows the same general approach as the proof of consistency for the sets of mono-Wunderlich curves discussed in Section 4.

Base case $\ell = 1$ As a base case, consider refinement level $\ell = 1$, and a face $F' = F_i^k$. Recall that f is defined based on subdivision of a d-dimensional unit hypercube—which we will denote U—into 2^d regions $S(0), ..., S((1)_d)$. This induces a subdivision of F' into 2^{d-1} regions $F'(0), ..., F'((1)_{d-1})$, each of which is a $(d-1)$-dimensional face of one of the regions $S(0), ..., S((1)_d)$. Let $F'(0), ..., F'((1)_{d-1})$ be indexed in the order in which the d-dimensional regions that contain them appear in $S(0), ..., S((1)_d)$. Thus, there is a monotonously increasing function $\sigma : \{0, ..., (1)_{d-1}\} \to \{0, ..., (1)_d\}$ such that $F'(r')$ is a face of $S(\sigma(r'))$. Note that for $r' \in \{0, ..., (1)_{d-1}\}$, the locations of the regions $F'(r')$ within U' are given by $\overset{i}{\times} c(\sigma(r')) = \overset{i}{\times} 2\text{RGC}^d(\sigma(r'))$. Furthermore, the co-domain of σ consists of exactly those values r such that $c_i(r) = 2\text{RGC}_i^d(r) = k$. Thus, the locations of the regions $F'(r')$ within U', in order of increasing r', form the sequence $\overset{i=k}{\times} 2\text{RGC}^d$, which, by Lemma 6, is exactly the sequence 2RGC^{d-1} if $k = 0$, and the sequence $\overset{i}{\ominus} 2\text{RGC}^{d-1}$ if $k = 1$ and $i < d-1$. Hence, the visible order of f on any face $F' = F_i^k$ visits the regions of refinement level $\ell = 1$ in the same order as f', or f' mirrored in dimension i, respectively, thus establishing the base case for our induction.

Induction For the inductive step, we have to be more careful than with the mono-Wunderlich curves. Before, all we had to do is proving that the transformations that map the curve f through U to the section of f through $S(\sigma(r'))$, also transform the order f' shown on a face $F^*(r')$ into the section of f' that should be visible on $F'(r')$. More concretely, we showed that we had to prove that we have $\overset{i}{\times} m(\sigma(r')) = m'(r')$ (Equation 1) and $\overset{i}{\times} a(\sigma(r')) = a'(r')$ (Equation 2). However, with the harmonious Hilbert curves, not all faces show f': some faces show a mirrored version, and one face (F_{d-1}^1) does not show f' in any way. Denote the face of U that maps to $F'(r')$ by $F^*(r')$. We will use r as a shorthand for $\sigma(r')$. Recall that we now have:
$$F^*(r') = F_{a_i(r)}^{|k - m_i(r)|}.$$

Therefore, we have to show:

- that we never have $k \ne m_i(r)$ while $a_i(r) = d-1$ (never use the 'broken' face);

- that the reflections $m(r)$ undo any mirroring of the order on $F^*(r')$ and apply any necessary mirroring on F';

- that the permutations match: $\overset{i}{\times} a(r) = a'(r')$ (this is no different from the situation with mono-Wunderlich curves in Section 4.4).

Recall that, by the definition of F' as F_i^k, we have $c_i(r) = k$.

We will first establish that we have $r' = \overset{i}{\times} r$. If $k = 0$, then, by definition of $r = \sigma(r')$, we have $c(r) = \overset{i=0}{\wedge} c'(r')$, and hence, $2\text{RGC}_i^d(r) = 0$ and $\overset{i}{\times} 2\text{RGC}^d(r) = 2\text{RGC}^{d-1}(r')$. By Lemma 5 we now have $r_i = r_{i-1}$, and thus, by Lemma 6, part (i), case (a), we have $r' = \overset{i}{\times} r$. Similarly, if $k = 1$ and $i < d-1$, then, by definition of $r = \sigma(r')$, we have $c(r) = \overset{i=1}{\wedge} \overset{i}{\ominus} c'(r')$, and hence, $2\text{RGC}_i^d(r) = 1$ and $\overset{i}{\times} 2\text{RGC}^d(r) = \overset{i}{\ominus} 2\text{RGC}^{d-1}(r')$. By Lemma 5 we now have $r_i \ne r_{i-1}$, and thus, by Lemma 6, part (i), case (c), we have $r' = \overset{i}{\times} r$.

Never the broken face   For the sake of contradiction, suppose $k \neq m_i(r)$ and $a_i(r) = d - 1$. We distinguish three cases: $r = 0$; $r > 0$ and $i < d - 1$; and $i = d - 1$.

First, consider the case of $r = 0$. Then $k = c_i(r)$ and $m_i(r) = c_i(r)$ by definition, so we cannot have $k \neq m_i(r)$.

Second, consider the case of $r > 0$ and $i < d - 1$. By the definition of the permutations $a(r)$ in harmonious Hilbert curves, it follows from $a_i(r) = d-1$ that $r_i = r_{d-1}$. However, by the definition of the reflections $m_i(r)$ in standard Hilbert curves, we also have $c_i(\check{r}) = m_i(r)$. Since $c_i(r) = k \neq m_i(r) = c_i(\check{r})$, this implies (by Lemma 3) that i is the smallest index such that $r_i \neq \check{r}_i$ and therefore $r_i = 1$ and $r_{d-1} = 0$. This contradicts the observation that $r_i = r_{d-1}$, so we cannot have $k \neq m_i(r)$ and $a_i(r) = d - 1$.

Third, consider the case of $i = d - 1$. In this case, our induction hypothesis only concerns the case $k = 0$, and thus we have $c_{d-1}(r) = 0$, $a_{d-1}(r) = d - 1$ and $m_{d-1}(r) = 1$. By the definition of the permutations $a(r)$ in harmonious Hilbert curves, $a_{d-1}(r) = d - 1$ if and only if $r_i \neq r_{d-1}$ for all $i < d - 1$. But then, by Lemma 5, $c_{d-1}(r) = |r_{d-1} - r_{d-2}| = 1$, which contradicts our observation that $c_{d-1}(r) = 0$. So in this case we cannot have $k \neq m_i(r)$ and $a_i(r) = d - 1$ either.

Reflections   We need to prove that the reflections $m(r)$ undo any mirroring of the order on $F^*(r')$ and apply any necessary mirroring on F'. Let j be $a_i(r)$ and let g be $\bar{a}_{j+1}(r)$, that is, g is defined such that $a_g(r) = a_i(r)+1$. We define $\mathring{m}(r)$ as the reflections of $S(r)$ with application (or equivalently, cancellation) of the mirroring of the order of f' as shown on $F^*(r')$. Note that the order of f' as shown on $F^*(r')$ is mirrored only if $m_i(r) \neq k$, and if so, it is mirrored in coordinate $a_i(r)$, which is coordinate $a_i(r) + 1$ in d-dimensional space on $F^*(r')$, and this coordinate is mapped to coordinate g on F'. Therefore, we have $\mathring{m}(r) = m(r)$ if $m_i(r) = k$, and $\mathring{m}(r) = \underset{\ominus}{\overset{g}{}} m(r)$ if $m_i(r) \neq k$. Furthermore, we define $\mathring{m}'(r')$ as the reflections of $F'(r)$ with the necessary mirroring of F', that is, $\mathring{m}'(r') = m'(r')$ if $k = 0$, and $\mathring{m}'(r') = \underset{\ominus}{\overset{i}{}} m'(r')$ if $k = 1$. What we need to prove is now: $\underset{\times}{\overset{i}{}} \mathring{m}(r) = \mathring{m}'(r')$.

We define $\mathring{c}$ as follows: if $m_i(r) = k$, then $\mathring{c} = c$; if $m_i(r) \neq k$, then $\mathring{c} = \underset{\ominus}{\overset{g}{}} c$. Thus, $\mathring{m}(r)$ is expressed in terms of $\mathring{c}(r)$ and $\mathring{c}(\check{r})$ just as $m(r)$ is expressed in terms of $c(r)$ and $c(\check{r})$ in the definition of standard Hilbert curves. Similarly, we define $\mathring{c}'$ as follows: if $k = 0$, then $\mathring{c}' = c'$; if $k = 1$, then $\mathring{c}' = \underset{\ominus}{\overset{i}{}} c'$. Thus, $\mathring{m}'(r')$ is expressed in terms of $\mathring{c}'(r')$ and $\mathring{c}'(\check{r}')$ just as $m(r)$ is expressed in terms of $c(r)$ and $c(\check{r})$ in the definition of standard Hilbert curves.

Note that $\underset{\wedge}{\overset{i=k}{}} \mathring{c}'$ gives the locations of the subregions of U' that touch F', in the order in which these subregions are visited by f; thus we have, for any $q' \in \{0, ..., (1)_{d-1}\}$:

$$\mathring{c}'(q') = \underset{\times}{\overset{i}{}} c(\sigma(q')). \tag{5}$$

We are now ready to prove $\underset{\times}{\overset{i}{}} \mathring{m}(r) = \mathring{m}'(r')$. We distinguish four cases:

(1) $m_i(r) = k$ and $r = 0$;

(2) $m_i(r) = k$ and $r > 0$ and $i < d - 1$;

(3) $m_i(r) = k$ and $r > 0$ and $i = d - 1$;

(4) $m_i(r) \neq k$ and $r' = 0$;

(5) $m_i(r) \neq k$ and $r' > 0$.

In case (1), we have $\mathring{m}(r) = m(r)$ (because $m_i(r) = k$), we have $c_i(r) = 0$ (because $r = 0$), and thus, $k = 0$ and $\mathring{m}'(r') = m'(r')$. Since $r' = \underset{\times}{\overset{i}{}} r = 0$ we have $\mathring{m}(r) = m(r) = c(r) = 0$ and $\mathring{m}'(r') = m'(r') = c'(r') = 0$, and therefore $\underset{\times}{\overset{i}{}} \mathring{m}(r) = \mathring{m}'(r')$, QED.

In case (2), we have $\mathring{m}(r) = m(r)$ (because $m_i(r) = k$). We have $c_i(\check{r}) = m_i(r) = k$. Thus, the subregion preceding $S(r)$ in the d-dimensional order also contributes the region preceding $F'(r')$ on F', of which the location is given by $\mathring{c}'(\check{r}')$ (note that if $k = 1$, then f' as visible on F' is mirrored in dimension i). Therefore we have $\mathring{c}'(\check{r}') = \underset{\times}{\overset{i}{}} c(\check{r})$ and $\mathring{c}'(r') = \underset{\times}{\overset{i}{}} c(r)$. Therefore, for $z < i$ we have: $\mathring{m}'_z(r') = \mathring{c}'_z(\check{r}') = c_z(\check{r}) = m_z(r)$; for $i \leq z < d - 2$ we have: $\mathring{m}'_z(r') = \mathring{c}'_z(\check{r}') = c_{z+1}(\check{r}) = m_{z+1}(r)$; and finally we have $\mathring{m}'_{d-2}(r') = 1 - \mathring{c}'_{d-2}(r') = 1 - c_{d-1}(r) = m_{d-1}(r)$. Thus, $\mathring{m}'(r') = \underset{\times}{\overset{i}{}} m(r) = \underset{\times}{\overset{i}{}} \mathring{m}(r)$, QED.

In case (3), by the definition of the permutations m in a standard Hilbert curve and the definition of the case $m_i(r) = k$, we have $c_{d-1}(r) = 1 - m_{d-1}(r) = 1 - k$, but this contradicts $c_i(r) = k$, so this cannot occur.

In case (4), we have $m_i(r) \neq k$, and therefore $\mathring{m}(r) = \underset{\ominus}{\overset{g}{}} m(r)$, where g is such that $a_g(r) = a_i(r) + 1$. Since $r' = 0$, we have that r equals the smallest q such that $c_i(q) = k$. If $k = 0$, this implies $r = 0$, but then, by the definition of reflections in a standard Hilbert curve, we would have $m_i(r) = c_i(r) = 0 = k$, contradicting the definition of case (4). Therefore, $k = 1$, and the induction hypothesis only concerns cases with $i < d - 1$. By Lemma 5, we now have $r = (0)_i 10(0)_{d-i-2}$ and $\check{r} = (0)_i 01(1)_{d-i-2}$. Applying the definitions of the permutations a and the reflections in m in standard and harmonious Hilbert curves, we find $a_i(r) = 0$, $g = \bar{a}_1(r) = d - 1$, $c(r) = (0)_i 11(0)_{d-i-2}$, $c(\check{r}) = (0)_i 01(0)_{d-i-2}$; if $i < d - 2$ we have $m(r) = (0)_i 01(0)_{d-i-3}1$; if $i = d - 2$ we have $m(r) = (0)_i 00$; therefore we have $\mathring{m}(r) = \underset{\ominus}{\overset{d-1}{}} m(r) = (0)_i 01(0)_{d-i-2}$ in all cases, and $\underset{\times}{\overset{i}{}} \mathring{m}(r) = (0)_i 1(0)_{d-i-2} = \underset{\ominus}{\overset{i}{}}(0)_{d-1} = \mathring{m}'(0)$, QED.

In case (5), we have $m_i(r) \neq k$, and therefore $\mathring{m}(r) = \underset{\ominus}{\overset{g}{}} m(r)$, where g is such that $a_g(r) = a_i(r) + 1$. Since $r' > 0$, we also have $r > 0$, and both $\check{r}$ and $\check{r}'$ exist. We have $c_i(\check{r}) = m_i(r) \neq k$. Thus, the subregion preceding $S(r)$ in the d-dimensional order does not contain the region preceding $F'(r')$ on F'. Let p' be $\check{r}'$ and let p be $\sigma(p')$. Note that we have $c_i(p) = k$, $c_i(\hat{p}), ..., c_i(\check{r}) = 1 - k$, $c_i(r) = k$, and:

$$\underset{\times}{\overset{i}{}} c(\check{r}) = \underset{\times}{\overset{i}{}} c(r). \tag{6}$$

By Lemma 3, as the rank increases from p to r, digit i must be the most significant digit that changes from 0 to 1 between p and $\hat{p}$, and the next time it changes from 0 to 1 is between $\check{r}$ and r. Therefore, between $\hat{p}$ and $\check{r}$, there is a unique more significant digit h that changes from 0 to 1, and it does so exactly once. Thus we get the following values for the digits of p, $\hat{p}$, $\check{r}$ and r:

x	$x_0, ..., x_{h-1}$	x_h	$x_{h+1}, ..., x_{i-1}$	x_i	$x_{i+1}, ..., x_{d-1}$
p	anything	0	1	0	1
$\hat{p}$	same as p	0	1	1	0
$\check{r}$	same as p	1	0	0	1
r	same as p	1	0	1	0

Note that we may have $h = i - 1$ or $i = d - 1$, so digits $x_{h+1}, ..., x_{i-1}$ and digits $x_{i+1}, ..., x_{d-1}$ might not exist. When rank digit h changes as the rank increases from p to r, the corresponding digit in the location changes as well. Thus, h must have the unique value such that $\mathring{c}'_h(p') \neq \mathring{c}'_h(r')$, and we get:

$$\mathring{c}'(p') = \underset{\ominus}{\overset{h}{}} \mathring{c}'(r') \tag{7}$$

Since $c_i(p) = c_i(r)$, we have that h also has the unique value such that $c_h(p) \neq c_h(r)$, so:

$$c(p) = \underset{\ominus}{\overset{h}{}} c(r) \tag{8}$$

Applying the definition of the permutations a in harmonious Hilbert curves to the numbers in the table, we find that we have $a_h(r) = a_i(r) + 1$. Thus, since $m_i(r) = 1 - k$, we have $\mathring{c} = \underset{\ominus}{\overset{h}{}} c$. Note that $p' = \underset{\times}{\overset{i}{}} p$ (just as $r' = \underset{\times}{\overset{i}{}} r$). Using Equations 5, 6, 7 and 8 and $h < i$, we get:

$$\mathring{c}'(\check{r}') = \mathring{c}'(p') = \underset{\ominus}{\overset{h}{}} \mathring{c}'(r') = \underset{\times}{\overset{i}{}} \underset{\ominus}{\overset{h}{}} c(r) = \underset{\times}{\overset{i}{}} \underset{\ominus}{\overset{h}{}} c(\check{r}) = \underset{\times}{\overset{i}{}} \mathring{c}(\check{r}). \tag{9}$$

Thus, for $z < i$ we have: $\mathring{m}'_z(r') = \mathring{c}'_z(\check{r}') = \mathring{c}_z(\check{r}) = \mathring{m}_z(r)$; for $i \leq z < d - 2$ we have $\mathring{m}'_z(r') = \mathring{c}'_z(\check{r}') = \mathring{c}_{z+1}(\check{r}) = \mathring{m}_{z+1}(r)$; and finally we have $\mathring{m}'_{d-2}(r') = 1 - \mathring{c}'_{d-2}(r') = 1 - c_{d-1}(r) = 1 - \mathring{c}_{d-1}(r) = \mathring{m}_{d-1}(r)$. Thus, $\mathring{m}'(r') = \underset{\times}{\overset{i}{}} \mathring{m}(r)$, QED.

Permutations Recall that we have $r' = \underset{\times}{\overset{i}{}} r$. Therefore, if $i < d - 1$, we have $r'_{d-2} = r_{d-1}$. If $i = d - 1$, the induction hypothesis only concerns the case of $k = 0$, that is, $c_{d-1}(r) = 0$, and therefore, by Lemma 5, we have $r_{d-1} = r_{d-2} = r'_{d-2}$. Thus, we always have $r'_{d-2} = r_{d-1}$.

Recall that the respective inverses of $a'(r')$ and $a(r)$ (where $r = \sigma(r')$) can be constructed from the identity permutation by moving all indices h such that $r'_h = r'_{d-2}$, or $r_h = r_{d-1} = r'_{d-2}$, respectively, to the front, reversing the order of those indices, and reversing the order of the remaining indices. Inserting a digit in r' at position i to obtain $r = \sigma(r')$, does not affect which of the other digits are moved to the front, since this only depends on the value of r'_{d-2}. Thus, inserting a digit in r' at position i to obtain r, moving the index i in $\bar{a}(r)$, and subsequently removing index i from $\bar{a}(r)$ (which is what $\underset{\times}{\overset{i}{}} a(r)$ does), does not affect the order of the remaining indices relative to each other. Therefore we have $a'(r') = \underset{\times}{\overset{i}{}} a(r)$, QED.

5.5 Orientation of the Butz-Moore and harmonious Hilbert curves

By Lemma 2, a d-dimensional mono-Hilbert curve has neutral orientation if and only if there is a number ℓ, such that each of the $d!$ possible permutations of d numbers can be constructed as the composition of ℓ permutations from a. Trivially, each 1-dimensional curve has neutral orientation. Below we analyse whether the d-dimensional curves for $d \geq 2$ have neutral orientation.

Butz-Moore curves   In the Butz-Moore curves, all permutations are rotations. Only if $d = 2$, this suffices to create all possible permutations by composition. Therefore, only the two-dimensional Butz-Moore curve, that is, only the original Hilbert curve, has neutral orientation; the higher-dimensional Butz-Moore curves have not.

Harmonious Hilbert curves   In the harmonious Hilbert curves, $a((0)_{d-2}10)$ is the permutation that swaps the first two elements and then reverses the whole sequence, and $a((0)_{d-i}(1)_i)$ is the permutation that rotates the sequence i steps to the right and then reverses the whole sequence. This allows us to construct any swap of two adjacent elements at positions $j-1$ and j by composing the four permutations $a((0)_{j+1}(1)_{d-j-1})$, $a((0)_{d-2}10)$, $a((0)_{d-j-1}(1)_{j+1})$ and $a((0)_d)$. Since any permutation can be created from any other by at most $d(d-1)/2$ swaps of adjacent elements, we get that any permutation can be created from any other by composing at most $2d(d-1)$ permutations from a. However, to satisfy the condition in Lemma 2, there should be a depth ℓ such that each permutation can be created from the identity permutation by composing *exactly* ℓ permutations. In the harmonious Hilbert curves, this can indeed be done for $\ell = 2d(d-1)$, since any permutation that is produced from $k < d(d-1)/2$ swaps can be produced by a sequence of $4k$ permutations producing the swaps, followed by $2d(d-1) - 4k$ reversal permutations $a((0)_d)$. It follows that the harmonious Hilbert curves have neutral orientation.

Note that the analysis given above is by no means tight: in reality much less than $2d(d-1)/2$ permutations from a may suffice to produce all possible permutations. For example, in four dimensions three permutations suffice, rather than $2d(d-1) = 24$.

6   First-half- and diagonal-extradimensional space-filling curves

6.1   Definition of first-half- and diagonal-extradimensionality

Recall that a d-dimensional space-filling curve f is *extradimensional to* a d'-dimensional space-filling curve f' with co-domain U', if the following holds for any valid ordering $\overline{f'}$ of f' and *any* choice of μ: there is a valid ordering $\overline{f}$ of f such that for any pair of points $a, b \in U'$ we have $\overline{f'}(a) < \overline{f'}(b)$ if and only if $\overline{f}(\text{lift}_\mu(a)) < \overline{f}(\text{lift}_\mu(b))$ (Definition 1). Here lift_μ is a function that defines the correspondence between points in the co-domains of f and f': the sequence of coordinates of $\text{lift}_\mu(p)$ is simply the sequence of coordinates of p, with zeros inserted in certain places according to μ. In this section we will consider two variants of extradimensionality: first-half-extradimensionality and diagonal-extradimensionality.

In the case of first-half-extradimensionality, $d = 2d'$ and μ is restricted to the identity function id, defined by $id(i) = i$. Thus, the coordinates of $\text{lift}_\mu(p)$ are simply the coordinates of p with d' zeros added at the end:

Definition 3 *A $2d'$-dimensional space-filling curve f is* first-half-extradimensional *to a d'-dimensional space-filling curve f' with co-domain U', if the following holds for any valid ordering $\overline{f'}$ of f': there is a valid ordering $\overline{f}$ of f such that for any pair of points $a, b \in U'$ we have $\overline{f'}(a) < \overline{f'}(b)$ if and only if $\overline{f}(\text{lift}_{id}(a)) < \overline{f}(\text{lift}_{id}(b))$.*

Given a d'-dimensional point $p = (p_0, ..., p_{d'-1})$, we define $\text{double}(p)$ as the $2d'$-dimensional point $(q_0, ..., q_{2d'-1})$ such that $q_i = p_{(i \bmod d')}$. In other words, $\text{double}(p)$ is obtained from p by concatenating two copies of the coordinate sequence of p. For a set of points P, let $\text{double}(P)$ be the set $\bigcup_{p \in P} \text{double}(p)$.

Definition 4 *A $2d'$-dimensional space-filling curve f is* diagonal-extradimensional *to a d'-dimensional space-filling curve f' with co-domain U', if the following holds for any valid ordering $\overline{f'}$ of f': there is a valid ordering $\overline{f}$ of f such that for any pair of points $a, b \in U'$ we have $\overline{f'}(a) < \overline{f'}(b)$ if and only if $\overline{f}(\text{double}(a)) < \overline{f}(\text{double}(b))$.*

An R-tree is a balanced tree structure in which the leaves store bounding boxes of d-dimensional objects. Each object is stored exactly once, and typically each leaf contains the bounding boxes of a fair number of objects: enough to fill a page on disk. As the bounding box of an object, the smallest axis-aligned d-dimensional box that contains the object is used. Each leaf or other node of the tree also has a bounding box of its own, which is the smallest axis-aligned d-dimensional box that encloses all objects stored in the subtree rooted at that node. In addition, each non-leaf node stores the bounding boxes of its children. This structure can be used to answer several types of queries efficiently [5], provided the object bounding boxes are distributed over the leaves in such a way that leaves have small bounding boxes.

One heuristic way to achieve this, is by the use of space-filling curves [11]. To determine the ordering of two-dimensional axis-parallel rectangles (object bounding boxes) in the tree, we represent the rectangles as four-dimensional points, which are subsequently sorted by the order of their positions along a four-dimensional space-filling curve. Two ways to represent rectangles $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ as four-dimensional points were considered. The *xy-representation* would use the point:

$$(x_{\min}, y_{\min}, x_{\max}, y_{\max}),$$

whereas the *cd-representation* (centre/dimensions) would use the point:

$$\left(\frac{x_{\min} + x_{\max}}{2}, \frac{y_{\min} + y_{\max}}{2}, x_{\max} - x_{\min}, y_{\max} - y_{\min} \right).$$

Since in practice, many rectangles are small, xy-representation leads to many points where the first and the third coordinate are almost identical, and the second and the fourth coordinate are almost identical. The other solution, cd-representation, leads to many points whose last two coordinates are almost zero. In previous work we argued and showed evidence that it is important that such four-dimensional points are ordered as much as possible as if the represented rectangles were simply ordered according to the positions of their centre points along a good two-dimensional space-filling curve [9]. This can be achieved by using four-dimensional curves that are diagonal- or first-half-extradimensional to Hilbert's two-dimensional curve.

The results in this paper could be used to try this approach to building R-trees in higher dimensions (for example, using six-dimensional curves for three-dimensional boxes), or to experiment with 3-regular curves instead of 2-regular curves for this application.

In previous work, we found that certain applications call for space-filling curves that are first-half- and/or diagonal-extradimensional to a given space-filling curve with good locality-preserving properties [9]. For details, see Inset 4.

Any $2d'$-dimensional Peano curve, coil curve, half-coil curve, Meurthe curve, or harmonious Hilbert curve (see Sections 4 and 5) is first-half-extradimensional, but generally not diagonal-extradimensional, to the d'-dimensional Peano curve, coil curve, half-coil curve, Meurthe curve, or harmonious Hilbert curve, respectively. In this section we will show how, given any d'-dimensional b-regular order-preserving vertex-continuous mono-curve f' with $b \in \{2, 3\}$ and $f'(0) = (0, 0, ..., 0)$, one can construct a $2d'$-dimensional b-regular vertex-continuous mono-curve f that is both first-half-extradimensional and diagonal-extra-dimensional to f'. This may provide a useful alternative to the curves from the previous sections if diagonal-extradimensionality is desired, or if one wants to base the $2d'$-dimensional curve on a d'-dimensional curve that is different from those presented in the previous sections.

6.2 Monotone space-filling curves

As a building block in our construction of first-half- and diagonal-extradimensional curves, we will use two-dimensional curves that have certain special properties. We will denote these curves by h_2 and h_3, where h_2 is the two-dimensional Hilbert curve (Figure 10, left), and h_3 is the Я-curve (Figure 10, right). Lemma 10 states that h_2 and h_3 have the properties which we will need to prove the correctness of our construction in Section 6.3.

Definition 5 *An ordering $\overline{f}$ of a space-filling curve f is* monotone from a to b, *if $\overline{f}(p)$ increases monotonically as a point p moves in a straight line from a to b.*

Lemma 10 *The curves h_2 and h_3 have valid orderings $\overline{h_2}$ and $\overline{h_3}$ that are monotone from $(0,0)$ to $(1,0)$ and from $(0,0)$ to $(1,1)$.*

Proof: We will prove the lemma for the Hilbert curve h_2; the proof for the Я-curve h_3 is very similar and is left as an exercise for the interested reader. In fact, we will prove a stronger claim for h_2, namely that h_2 has a valid ordering $\overline{h_2}$ that is monotone from $(0,0)$ to $(0,1)$, from $(0,1)$ to $(1,1)$, from $(1,1)$ to $(1,0)$, from $(0,0)$ to $(1,0)$, and from $(0,0)$ to $(1,1)$.

rank	loc.	perm.	refl.	exit
00	00	10	00	$\frac{1}{2}(0,1)$
01	01	01	00	$\frac{1}{2}(1,1)$
10	11	01	00	$\frac{1}{2}(2,1)$
11	10	10	11	$\frac{1}{2}(2,0)$

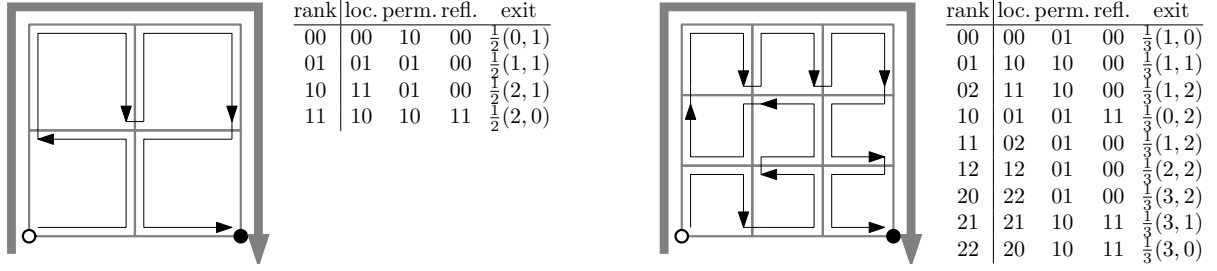| rank	loc.	perm.	refl.	exit
00	00	01	00	$\frac{1}{3}(1,0)$
01	10	10	00	$\frac{1}{3}(1,1)$
02	11	10	00	$\frac{1}{3}(1,2)$
10	01	01	11	$\frac{1}{3}(0,2)$
11	02	01	00	$\frac{1}{3}(1,2)$
12	12	01	00	$\frac{1}{3}(2,2)$
20	22	01	00	$\frac{1}{3}(3,2)$
21	21	10	11	$\frac{1}{3}(3,1)$
22	20	10	11	$\frac{1}{3}(3,0)$

Figure 10: Hilbert's curve (left) and the Я-curve (right), defined graphically and by means of a table.

We will prove the lemma by induction on increasing level of refinement of the subdivision of the unit square into subsquares. More precisely, for a level of refinement ℓ, we consider the recursive balanced subdivision of the unit square into 4^ℓ square regions, and prove the following induction hypothesis: there is a valid ordering $\overline{h_2}$ that:

(1) puts the regions touching the left edge of the unit square in order from bottom $(0,0)$ to top $(0,1)$;

(2) puts the regions touching the top edge of the unit square in order from left $(0,1)$ to right $(1,1)$;

(3) puts the regions touching the right edge of the unit square in order from top $(1,1)$ to bottom $(1,0)$;

(4) puts the regions touching the bottom edge of the unit square in order from left $(0,0)$ to right $(1,0)$;

(5) puts the regions on the ascending diagonal of the unit square in order from bottom $(0,0)$ to top $(1,1)$.

As a base case, we take $\ell = 1$. For this base case, the induction hypothesis can easily be verified by inspection of Figure 10.

We will now prove that if the induction hypothesis holds for refinement level $\ell - 1$, then it also holds for refinement level ℓ. Note that, as far as the left edge, the bottom edge and the diagonal are concerned, the induction hypothesis is invariant under the transformation applied in the first (lower left) quadrant of h_2, and, as far as the right and the bottom edge are concerned, the induction hypothesis is invariant under the transformation applied in the last (lower right) quadrant of h_2. Thus $\overline{h_2}$ places:

- the regions touching the lower half of the left edge of the unit square before the regions touching the upper half of the left edge of the unit square (by the induction hypothesis for level 1);

- the regions touching the lower half of the left edge of the unit square in order from bottom to top (by the induction hypothesis for level $\ell - 1$);

- the regions touching the upper half of the left edge of the unit square in order from bottom to top (by the induction hypothesis for level $\ell - 1$).

As a result, $\overline{h_2}$ puts all level-ℓ regions touching the left edge of the unit square in order from bottom to top. In a similar way, one can verify the order induced by $\overline{h_2}$ on the regions touching the top, right, and bottom edges and on the ascending diagonal of the unit square. $\qquad\square$

6.3 Constructing first-half- and diagonal-extradimensional curves

To be able to present and analyse our construction, we need some notation. When $f : [0,1] \to [0,1]^d$ is a d-dimensional space-filling curve, let $f^{[i]}(t)$ be coordinate i of $f(t)$, where coordinates are numbered from 0 to $d-1$. In other words, $f(t) = (f^{[0]}(t), f^{[1]}(t), ..., f^{[d-1]}(t))$. For two d'-dimensional points p and q, we let $p|q$ be the $2d'$-dimensional point obtained by concatenating the coordinate sequences of p and q, that is, $p|q = (p_0, ..., p_{d'-1}, q_0, ..., q_{d'-1})$. For a $2d'$-dimensional point p, we let $p^\dashv$ denote the d'-dimensional point obtained by taking the first d' coordinates of p, that is, $p^\dashv = (p_0, ..., p_{d'-1})$. Similary, $p^\vdash$ is the point obtained by taking the last d' coordinates of p, that is, $p^\vdash = (p_{d'}, ..., p_{2d'-1})$. We use 0 not only to denote the number zero, but also to denote the origin of the coordinate system, in any number of dimensions.

Now, given a d'-dimensional b-regular space-filling curve f' with $b \in \{2,3\}$, we can define a $2d'$-dimensional space-filling curve f by:

$$f(t) = f'(h_b^{[0]}(t)) \,|\, f'(h_b^{[1]}(t))$$

To put it differently: for $k \in \{0,1\}$ and $i \in \{0, ..., d'-1\}$, we define $f^{[kd'+i]}(t)$ as $f'^{[i]}(h_b^{[k]}(t))$. With a slight abuse of terminology and notation, we can think of f as the composition of f' and h_b, denoted $f' \circ h_b$. Below we will prove the following:

Theorem 3 *If f' is an order-preserving vertex-continuous mono-curve and $f'(0) = 0$, then $f = f' \circ h_b$ is a b-regular vertex-continuous space-filling curve, and f is first-half-extradimensional and diagonal-extradimensional to f'. If, in addition, f' is symmetric, then f is an order-preserving mono-curve.*

In Section 6.4 we will see how to implement a comparison operator for f. We will now prove Theorem 3 in four parts (Lemmas 11–14).

Lemma 11 *If f' is a order-preserving b-regular vertex-continuous mono-curve, then $f = f' \circ h_b$ is a b-regular vertex-continuous space-filling curve.*

Proof: For each point $p \in [0,1]^{2d'}$ there are values $a, b \in [0,1]$ such that $f'(a) = p^{\dashv}$ and $f'(b) = p^{\vdash}$ (because f' is a surjective function from $[0,1]$ to $[0,1]^{d'}$), and there is a value t such that $h_2(t) = (a, b)$ (because h_2 is a surjective function from $[0,1]$ to $[0,1]^2$). Therefore f is a surjective function from $[0,1]$ to $[0,1]^{2d'}$. Furthermore, since f' and h_b are continuous, so is f.

It remains to prove that f is b-regular. Consider level $d'\ell$ of refinement of h_b. At this level of refinement, h_b maps each interval $[r/b^{2d'\ell}, (r+1)/b^{2d'\ell}]$, for $r \in \{0, ..., b^{2d'\ell} - 1\}$, to a square of width $1/b^{d'\ell}$. The horizontal range of the square, that is, the possible values of $h_b^{[0]}$ that are attained in this square, correspond exactly to the pre-image of one of the $b^{d'\ell}$ level-ℓ regions of f', thus defining the region occupied in $2d'$-dimensional space with respect to the first d' dimensions. Similarly, the vertical range of the square corresponds exactly to the pre-image of a region of f', thus defining the region occupied in $2d'$-dimensional space with respect to the last d' dimensions. Thus, each interval $[r/b^{2d'\ell}, (r+1)/b^{2d'\ell}]$ is mapped exactly to the Cartesian product of two d'-dimensional hypercube regions of width $1/b^{\ell}$, which is a $2d'$-dimensional hypercube region of width $1/b^{\ell}$. This implies the hierarchical structure of a b-regular curve. □

Lemma 12 *If f' is a symmetric b-regular mono-curve, then $f = f' \circ h_b$ is an order-preserving b-regular mono-curve.*

Proof: Extending our notation from Section 2.3, we denote the subregions, locations, permutations, and reflections of a curve f by S^f, c^f, a^f and m^f, respectively. Let $h_{b.\ell}$ be the level ℓ of refinement of h_b, at which level regions are numbered from 0 to $b^{2\ell} - 1$, and the region coordinates $c_i^{h_{b.\ell}}$ are between 0 and $b^\ell - 1$ (inclusive). The transformation of f in region $r \in \{0, ..., b^{2d'} - 1\}$ is the result of the successive effects of (i) the permutation $a^{h_{b.d'}}(r)$, (ii) the reflections $m^{h_{b.d'}}(r)$, (iii) the permutations $a^{f'}(c_k^{h_{b.d'}}(r))$ for $k \in \{0,1\}$, and (iv) the reflections $m^{f'}(c_k^{h_{b.d'}}(r))$. Note that there are no reversals, since f' is symmetric, and therefore, order-preserving.

(i) The permutation $a^{h_{b.d'}}(r)$ is either 01 or 10: the latter results in a rotation of f that consists in swapping coordinates $0, ..., d'-1$ with coordinates $d', ..., 2d'-1$.

(ii) Because f' is symmetric, there is a transformation ρ (consisting of rotation and/or reflections) such that $f'(1 - h_b^{[k]}(t)) = \rho(f'(h_b^{[k]}(t)))$. A reflection $m_0^{h_{b.d'}}(r)$ results in replacing the coordinates $f^{\dashv}(t) = f'(h_b^{[0]}(t))$ by $f'(1 - h_b^{[0]}(t)) = \rho(f'(h_b^{[0]}(t)) = \rho(f^{\dashv}(t))$ for $i \in \{0, ..., d-1\}$, which results in a transformation of f that consists in applying ρ to the first d' coordinates while leaving the last d' coordinates unaffected; thus, this transformation of f consists of rotations and reflections in $2d'$-dimensional space. Similarly, a reflection $m_1^{h_{b.d'}}(r)$ results in a transformation of f that consists in applying ρ to the last d' coordinates while leaving the first d' coordinates unaffected.

(iii,iv) Similar to ρ, the permutations $a^{f'}(c_k^{h_{b.d'}}(r))$ and the reflections $m^{f'}(c_k^{h_{b.d'}}(r))$ result in transformations of f that consist of rotations and reflections.

Thus, the transformation of f in region $r \in \{0, ..., b^{2d'} - 1\}$ consists of rotations and reflections, and f is an order-preserving mono-curve. □

Table 3: The four-dimensional curve $h_2 \circ h_2$, created from the composition of two-dimensional Hilbert curves. This curve is first-half- and diagonal-extradimensional to the two-dimensional Hilbert curve.

rank	loc.	perm.	refl.	exit	rank	loc.	perm.	refl.	exit
0000	0000	1032	0000	$\frac{1}{2}(0,1,0,0)$	1000	1111	2301	0000	$\frac{1}{2}(1,1,2,1)$
0001	0100	2310	0000	$\frac{1}{2}(0,1,0,1)$	1001	1110	0132	0011	$\frac{1}{2}(2,1,2,1)$
0010	0101	2301	0000	$\frac{1}{2}(0,1,1,1)$	1010	1010	1032	1111	$\frac{1}{2}(2,0,2,1)$
0011	0001	1023	0110	$\frac{1}{2}(0,0,1,1)$	1011	1011	3201	1010	$\frac{1}{2}(2,0,1,1)$
0100	0011	3201	0000	$\frac{1}{2}(0,0,2,1)$	1100	1001	1023	1010	$\frac{1}{2}(2,1,1,1)$
0101	0010	1032	0011	$\frac{1}{2}(0,1,2,1)$	1101	1101	2301	1010	$\frac{1}{2}(2,1,0,1)$
0110	0110	0132	0011	$\frac{1}{2}(1,1,2,1)$	1110	1100	2310	1001	$\frac{1}{2}(2,1,0,0)$
0111	0111	2301	1010	$\frac{1}{2}(1,1,1,1)$	1111	1000	1032	1100	$\frac{1}{2}(2,0,0,0)$

Note that if f' is not symmetric, then f is not necessarily a mono-curve. A reflection of h_b in coordinate $k \in \{0,1\}$ results in replacing $f^{[kd'+i]}(h_b^{[k]}(t))$ by $f^{[kd'+i]}(1 - h_b^{[k]}(t))$ for $i \in \{0,...,d'-1\}$. In general, this transformation cannot be expressed as a simple permutation, reflection, or reversal of the d-dimensional curve f. For example, if $b = 3$ and $f' = h_3$ (the Я-curve of Figure 10), then f is not a mono-curve. The same might be the case if f' is a Meurthe curve (see Section 4).

However, if f' is symmetric (for example, a Butz-Moore Hilbert curve or a harmonious Hilbert curve from Section 5, or a Peano curve, a coil curve, or a half-coil curve from Section 4), then f will be an order-preserving mono-curve. For example, if f' is a standard Hilbert curve, then (using the notation from the proof of Lemma 12) a reflection $m_k^{h_b \cdot d'}(r)$ leads to a reflection in axis $\overline{a}_0^{f'}(c_k^{h_b \cdot d'}(r))$. Thus we get:

$$
\begin{aligned}
c_{kd'+i}^{f}(r) &= c_i^{f'}(c_k^{h_2 \cdot d'}(r)) \\
a_{kd'+i}^{f}(r) &= a_i^{f'}(c_k^{h_2 \cdot d'}(r)) + d' \cdot a_k^{h_2 \cdot d'}(r) \\
m_{kd'+i}^{f}(r) &= m_i^{f'}(c_k^{h_2 \cdot d'}(r)) && \text{if } m_k^{h_2 \cdot d'}(r) = 0 \text{ or } a_i^{f'}(c_k^{h_2 \cdot d'}(r)) \neq 0 \\
m_{kd'+i}^{f}(r) &= 1 - m_i^{f'}(c_k^{h_2 \cdot d'}(r)) && \text{if } m_k^{h_2 \cdot d'}(r) = 1 \text{ and } a_i^{f'}(c_k^{h_2 \cdot d'}(r)) = 0
\end{aligned}
$$

Table 3 shows the resulting curve definition if f' is the two-dimensional Hilbert curve, and thus, the resulting curve f is $h_2 \circ h_2$. Note that f is not a standard Hilbert curve as defined in Section 5.1, since the locations of the subregions do not conform to the binary reflected Gray code.

Lemma 13 *The curve f is first-half-extradimensional to f'.*

Proof: Let $\overline{f'}$ be any valid ordering of f. By Definition 3, we have to prove that there is a valid ordering $\overline{f}$ of f such that for any pair of points $a, b \in U'$ we have $\overline{f'}(a) < \overline{f'}(b)$ if and only if $\overline{f}(a|0) < \overline{f}(b|0)$.

Let $\overline{h_b}$ be a valid ordering of h_b which is monotone from $(0,0)$ to $(1,0)$ (such an ordering exists by Lemma 10). We now define $\overline{f}$ by $\overline{f}(p) = \overline{h_b}(\overline{f'}(p^\dashv), \overline{f'}(p^\vdash))$. Recall that f' is a b-regular curve with the entrance gate in the origin, and therefore $\overline{f'}(0) = 0$. Thus, we have $\overline{f}(a|0) = \overline{h_b}((\overline{f'}(a), \overline{f'}(0))) = \overline{h_b}((\overline{f'}(a), 0))$ and $\overline{f}(b|0) = \overline{h_b}((\overline{f'}(b), \overline{f'}(0))) = \overline{h_b}((\overline{f'}(b), 0))$. By the monotonicity of $\overline{h_b}$ from $(0,0)$ to $(1,0)$, it follows that $\overline{f}(a|0) < \overline{f}(b|0)$ if and only if $\overline{f'}(a) < \overline{f'}(b)$, QED. □

Lemma 14 *The curve f is diagonal-extradimensional to f'.*

Proof: Let $\overline{f'}$ be any valid ordering of f. By Definition 4, we have to prove that there is a valid ordering $\overline{f}$ of f such that for any pair of points $a, b \in U'$ we have $\overline{f'}(a) < \overline{f'}(b)$ if and only if $\overline{f}(a|a) < \overline{f}(b|b)$.

Let $\overline{h_b}$ be a valid ordering of h_b which is monotone from $(0,0)$ to $(1,1)$ (such an ordering exists by Lemma 10). We now define $\overline{f}$ by $\overline{f}(p) = \overline{h_b}(\overline{f'}(p^\dashv), \overline{f'}(p^\vdash))$. Thus, we have $\overline{f}(a|a) = \overline{h_b}((\overline{f'}(a), \overline{f'}(a)))$ and $\overline{f}(b|b) = \overline{h_b}((\overline{f'}(b), \overline{f'}(b)))$. By the monotonicity of $\overline{h_b}$ from $(0,0)$ to $(1,1)$, it follows that $\overline{f}(a|a) < \overline{f}(b|b)$ if and only if $\overline{f'}(a) < \overline{f'}(b)$, QED. □

6.4 Implementing a first-half- and diagonal-extradimensional curve

In Section 6.3 we saw how to one can construct a first-half- and diagonal-extradimensional curve f, given a b-regular order-preserving vertex-continuous mono-curve f' that satisfies certain conditions. We also saw that if f' is symmetric, then f is an order-preserving mono-curve, which could be implemented using Algorithm 2.2.

We will now see another method to implement a comparison operator for f, which can also be applied if f' is not symmetric and which may also be more efficient. We illustrate this method by implementing first-half- and diagonal-extradimensional curves based on the Butz-Moore generalization of Hilbert curves. Our goal is to compare points according to a valid ordering $\overline{f}$ of f. Considering the construction of f, we can implement this by computing $\overline{f}(p)$ as $\overline{h_2}(\overline{f'}(p^\dashv), \overline{f'}(p^\vdash))$. To do so, we rewrite Algorithm 5.1 as a comparator object for Butz-Moore curves that is initialized with the coordinates of two d'-dimensional points p' and q', and provides a method to compute $\overline{f'}(p')$ and $\overline{f'}(q')$ one digit a time. A comparison operator for $\overline{f}$ is then implemented by initializing two comparator objects, one with $p^\dashv$ and $q^\dashv$, and one with $p^\vdash$ and $q^\vdash$, and then simply computing $\overline{h_2}$, extracting digits from the comparator objects rather than directly from the coordinates of p and q.

The details are as follows. Each object has member variables *forward*, *rotation*, *reflected*, i, *iMinusOne*, *newRotation*, *pLocal* and *qLocal*. To initialize an object, we run Algorithm 6.1, calling it with two d'-dimensional points p and q; to extract a rank digit (one for p and one for q), we run Algorithm 6.2. The extraction method works correctly up to and including the first time the rank digit differs between p and q (digits extracted afterwards may be wrong). The algorithm for the $2d'$-dimensional curve is given as Algorithm 6.3.

Algorithm 6.1: Initialization of a comparator object for a Butz-Moore curve

---

1 $pLocal \leftarrow p$; $qLocal \leftarrow q$; $forward \leftarrow$ **true**; $rotation \leftarrow 0$; **for** $i \leftarrow 0$ **to** $d' - 1$ **do** $reflected[i] \leftarrow$ **false**
2 $i \leftarrow rotation$; $iMinusOne \leftarrow d' - 1$; $newRotation \leftarrow iMinusOne$

Algorithm 6.2: Extracting a single rank digit from a comparator object for a Butz-Moore curve

---

1 $pDigit \leftarrow \textsc{ExtractFirstDigit}(pLocal[i])$; $qDigit \leftarrow \textsc{ExtractFirstDigit}(qLocal[i])$
2 **if** $reflected[i]$ **then**
3 $\quad$ $pDigit \leftarrow (1 - pDigit)$; $qDigit \leftarrow (1 - qDigit)$
4 **if** $forward$ **then**
5 $\quad$ $pRank \leftarrow pDigit$; $qRank \leftarrow qDigit$
6 **else**
7 $\quad$ $pRank \leftarrow (1 - pDigit)$; $qRank \leftarrow (1 - qDigit)$
8 **if** $pDigit = 1$ **then**
9 $\quad$ $forward \leftarrow \neg forward$; $reflected[i] \leftarrow \neg reflected[i]$; $newRotation \leftarrow iMinusOne$
10 $iMinusOne \leftarrow i$; $i \leftarrow (i + 1) \bmod d'$
11 **if** $i = rotation$ **then**
12 $\quad$ $reflected[iMinusOne] \leftarrow \neg reflected[iMinusOne]$
13 $\quad$ $reflected[newRotation] \leftarrow \neg reflected[newRotation]$
14 $\quad$ $rotation \leftarrow newRotation$
15 $\quad$ $i \leftarrow rotation$; $iMinusOne \leftarrow (i - 1) \bmod d'$; $newRotation \leftarrow iMinusOne$
16 **return** $(pRank, qRank)$

---

7   Remaining questions for theory and practice

Building R-trees   As explained in Inset 4, the curves in this paper could be used to build R-trees in higher dimensions (for example, using six-dimensional curves for three-dimensional boxes), or to experiment with 3-regular curves instead of 2-regular curves for this application. Some first, rough experiments, indicate that it may depend a lot on the distribution of the data whether the new curves offer any advantages over previously known curves—but it seems unlikely that the new curves have serious disadvantages in this application. Further experiments would need to be done.

Fast implementations   To get most out of the space-filling curves presented in this paper, one may want to design highly-optimized implementations of comparison operators. As compared to the 3-

Algorithm 6.3: Algorithm to compare the position of two points p and q along a $2d'$-dimensional curve that is first-half- and diagonal-extradimensional to a d'-dimensional Butz-Moore curve.

1 initialize a comparator object $compare[0]$ with points $p^{\daleth}$ and $q^{\daleth}$
2 initialize a comparator object $compare[1]$ with points $p^{\vdash}$ and $q^{\vdash}$
3 $forward \leftarrow \mathbf{true}$; $rotation \leftarrow 0$; **for** $i \leftarrow 0$ **to** 1 **do** $reflected[i] \leftarrow \mathbf{false}$
4 **repeat**
5 $i \leftarrow rotation$; $iMinusOne \leftarrow (i-1) \bmod 2$; $newRotation \leftarrow iMinusOne$
6   **repeat**
7   extract $(pDigit, qDigit)$ from $compare[i]$
8   **if** $reflected[i]$ **then**
9 $pDigit \leftarrow (1 - pDigit)$; $qDigit \leftarrow (1 - qDigit)$
10   **if** $pDigit < qDigit$ **then return** $forward$; **else if** $pDigit > qDigit$ **then return** $\neg forward$
11   **if** $pDigit = 1$ **then**
12 $forward \leftarrow \neg forward$; $reflected[i] \leftarrow \neg reflected[i]$; $newRotation \leftarrow iMinusOne$
13 $iMinusOne \leftarrow i$; $i \leftarrow (i+1) \bmod 2$
14   **until** $i = rotation$
15 $reflected[iMinusOne] \leftarrow \neg reflected[iMinusOne]$
16 $reflected[newRotation] \leftarrow \neg reflected[newRotation]$
17 $rotation \leftarrow newRotation$
18 **until** all remaining digit strings $p[0], ..., p[2d'-1]$ and $q[0], ..., q[2d'-1]$ are empty
19 **return false**

regular curves, the 2-regular curves have the advantage that many calculations can be implemented by means of bit shifting and/or exclusive-or operations on many bits at once. A possible advantage of interdimensionally consistent curves, is that a highly-optimized (possibly hard-wired) implementation of a d-dimensional curve, for some fixed value of d, can also be used as an implementation of each d'-dimensional curve with $d' \leq d$. If the cost of working with more dimensions than necessary is mitigated by parallel circuitry, then, being able to use a single hard-wired d-dimensional implementation for any $d' \leq d$, may reduce overhead.

Proving that the harmonious Hilbert curves are interdimensionally consistent At this point, I would like to pose the main result of this paper as an open question. In Section 5.4, I proved that the harmonious Hilbert curves consitute an interdimensionally consistent set. However, I believe the proof is hard to understand and to verify. In particular, I do not find the current proof more convincing than several earlier attempts at a proof, which I only dismissed as erroneous because they led to the conclusion that the harmonious Hilbert curves could not be interdimensionally consistent. Indeed, these earlier attempts had to be wrong, or my exhaustive search algorithm to find the curves and the algorithms I used to test the implementations all had to be wrong. I implemented comparison operators for all curves from Section 4 and 5 for $d \in \{2, 3, 4, 5\}$ and verified interdimensional consistency with respect to the ordering of subregions down to at least the third level of refinement. Although now, everything seems to fall into place and I am convinced that my implementations and Section 5.4 do not contain any mistakes that cannot be repaired, it would still be good if we could find a simpler proof.

Clearing up interdimensional consistency under tie-breaking Tie-breaking is a subtle issue which was touched on in Sections 1 and 2, but not resolved completely. Space-filling curves are surjective, but not bijective—in fact, they cannot be bijective. Often this issue is dealt with by considering the level-ℓ approximation of the space-filling curve—say, the Hilbert curve h_2, for concreteness—which is the polygonal curve whose vertices are the lower left corners of the 4^ℓ successive squares at the ℓ-th level of refinement. Clearly, for any ℓ, the level-ℓ approximation visits each point of the unit square at most once, and in the limit for $\ell \to \infty$, the approximation visits all points of the unit square. Unfortunately, this does not imply that each point is visited only once: this approach offers no way around the fact that, for example, $h_2(1/6) = h_2(1/2)$. This can be seen by writing out $h_2(1/6)$ and $h_2(1/2)$ in binary notation:

$h_2(1/110) = h_2(0.00101010...) = (0.0111..., 0.0111...) = (0.1, 0.1)$, and $h_2(1/10) = h_2(0.1) = (0.1, 0.1)$. This makes the inverse of h_2 ill-defined.

In the definitions and proofs in this paper, we dealt with the ambiguity as follows. We allow any 'inverse' $\overline{h_2}$ such that for all points p in the unit hypercube, we have $h_2(\overline{h_2}(p)) = p$; and we prove that no matter which inverse $\overline{h_2}$ we choose, we can choose at least one valid inverse for each higher-dimensional harmonious Hilbert curve so that we get an interdimensionally consistent set.

However, our implementations of comparison operators are essentially based on level-ℓ approximations. The idea is that this implements the tie-breaking method that always assigns each point p to the quadrant that lies to its upper right. In fact, this approach would not work correctly if we would accept coordinates that end with an infinite sequence 111...: it would put the points $(0.0111, 0.0111)$, $(0, 1)$ and $(0.1, 0.1)$ in this order, even though the first and the last point are the same. Our algorithms do work correctly as long as the coordinates of each point p are represented as binary numbers with a *finite* number of digits. Recall that our algorithms do not accept the number 1 for a coordinate.

The above observations imply that the proofs of interdimensional consistency for the curves as such, are not necessarily valid for the curves with the specific choice of inverses as implemented in our algorithms. The fact that our implementations of mono-Wunderlich curves are interdimensionally consistent could still be established easily by inspecting the algorithms (see Section 4.2). For harmonious Hilbert curves the situation is less clear. The algorithm for harmonious Hilbert curves is much harder to analyse. The proof that harmonious Hilbert curves are interdimensionally consistent relies on the proper handling of points with coordinates equal to 1, but the implementation does not accept the number 1 and has not been verified to work correctly when given the alternative representation 0.111... of the same number. Although everything seems to indicate that this is ultimately just a matter of presentation, it is one that would be good to resolve.

Interdimensionally consistent sets of mono-Hilbert curves with interior origins? In Section 5.2 I had to give up on extending the harmonious Hilbert curves to filling the full d-dimensional space (rather than only the space of points with non-negative coordinates) while retaining interdimensional consistency. The reason was that I could not find a way to move the origin into the interior of the d-dimensional unit hypercube, while maintaining that the d-dimensional harmonious Hilbert curve shows the $(d-1)$-dimensional curve on the intersection of the unit hypercube with any axis-parallel $(d-1)$-dimensional plane through the origin. I conjecture that this is, in fact, impossible. One could also try to find out if an interdimensionally consistent set of 2-regular vertex-continuous space-filling curves—not necessarily mono-curves—could exist.

How unique are sets of interdimensionally consistent curves? The fact that the three-, four-, five-, six- and seven-dimensional harmonious Hilbert curves emerged as the unique results of an automated search, leads me to conjecture that the harmonious Hilbert curves actually constitute the *only* interdimensionally consistent set of mono-Hilbert curves. The catch is that, for d-dimensional curves with $d \geq 4$, the search was restricted to symmetric curves in which the subregions are traversed in the order of binary reflected Gray codes. Although this seems to be the most natural choice, and although this is indeed the only solution for $d = 2$ and $d = 3$, I cannot be sure that no alternative solutions for $d \geq 4$ were ruled out by these restrictions.

While I believe that the mono-Hilbert curves may contain only one interdimensionally consistent set, we already saw that the mono-Wunderlich curves contain at least four interdimensionally consistent sets. There must be many more, if only because the subregion in the centre can be rotated freely without affecting the curves shown on the faces of the unit hypercube. One could try to find a full characterization of all interdimensionally consistent sets of mono-Wunderlich curves—possibly with the added condition that the back faces also show the $(d-1)$-dimensional curves.

Acknowledgements

References

[1] J. Alber and R. Niedermeier. On multidimensional curves with Hilbert property. *Theory of Computing Systems*, 33(4):295–312 (2000)

[2] M. Bader. *Space-filling curves: an introduction with applications in scientific computing.* Springer, 2013.

[3] A. R. Butz. Alternative algorithm for Hilbert's space-filling curve. *IEEE Trans. Comp.*, April, 1971, pp 424–426.

[4] M. Gardner. Mathematical Games—In which "monster" curves force redefinition of the word "curve". *Scientific American*, 235(6):124–133, 1976.

[5] H. Haverkort. Introduction to bounding volume hierarchies. Excerpt from Chapter 1 of *Results on Geometric Networks and Data Structures.* PhD thesis, Universiteit Utrecht, 2004.

[6] H. Haverkort. *An inventory of three-dimensional Hilbert space-filling curves.* arXiv:1109.2323.

[7] H. Haverkort. Recursive tilings and space-filling curves with little fragmentation. *Journal of Computational Geometry* 2(1): 92-127 (2011)

[8] H. Haverkort and F. van Walderveen. Locality and bounding-box quality of two-dimensional space-filling curves. *Computational Geometry*, 43(2):131–147 (2010)

[9] H. Haverkort and F. van Walderveen. Four-dimensional Hilbert curves for R-trees. *ACM Journal of Experimental Algorithmics*, 16 (2011)

[10] D. Hilbert. Über die stetige Abbildung einer Linie auf ein Flächenstück. *Math. Ann.*, 38(3):459–460 (1891)

[11] I. Kamel and C. Faloutsos. On packing R-trees. In *Conf. on Information and Knowledge Management* 1993, pp 490–499.

[12] H. L. Lebesgue. *Leçons sur l'intégration et la recherche des fonctions primitives*, pp 44–45. Gauthier-Villars, 1904.

[13] U. von Luxburg. *Lokalitätsmaße von Peanokurven.* Student project report, Universität Tübingen, Wilhelm-Schickard-Institut für Informatik, 1998.

[14] D. Moore. *Fast hilbert curve generation, sorting, and range queries.* http://web.archive.org/web/www.caam.rice.edu/~dougm/twiddle/Hilbert/, 2000.

[15] G. Peano. Sur une courbe, qui remplit toute une aire plane. *Math. Ann.*, 36(1):157–160 (1890)

[16] J.-M. Wierum. *Definition of a new circular space-filling curve: $\beta\Omega$-indexing.* Technical Report TR-001-02, Paderborn Center for Parallel Computing (PC^2), 2002.

[17] W. Wunderlich. Über Peano-Kurven. *Elemente der Mathematik*, 28(1):1–10, 1973.